TUGAS AKHIR – TE 141599

DESAIN KONTROLER *FUZZY* PID *GAIN SCHEDULING* UNTUK PENGATURAN KECEPATAN MOTOR DC TANPA SIKAT

Hudaibiy Hibban
NRP. 2211 100 108

Dosen Pembimbing
Ir. Josaphat Pramudijanto, M.Eng.
Nurlita Gamayanti ST., MT.

JURUSAN TEKNIK ELEKTRO
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
Surabaya 2015

FINAL PROJECT – **TE 141599**

DESIGN FUZZY PID GAIN SCHEDULING CONTROLLER FOR BRUSHLESS DC MOTOR SPEED CONTROL

Hudaibiy Hibban
NRP. 2211 100 108

Supervisor
Ir. Josaphat Pramudijanto, M.Eng.
Nurlita Gamayanti S.T., M.T.

DEPARTEMENT OF ELECTRICAL ENGINEERING
Faculty of Industrial Technology
Sepuluh Nopember Insitute of Technology
Surabaya 2015

DESAIN KONTROLER *FUZZY* PID *GAIN SCHEDULING* UNTUK PENGATURAN KECEPATAN MOTOR DC TANPA SIKAT

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Teknik
Pada
Bidang Studi Teknik Sistem Pengaturan
Jurusan Teknik Elektro
Fakultas Teknik Industri
Institut Teknologi Sepuluh Nopember

Mengetahui / Menyetujui

Dosen Pembimbing I,	Dosen Pembimbing II,
Ir. Josaphat Pramudijanto, M.Eng.	**Nurlita Gamayanti ST., MT.**
NIP. 19621005199003 1003	**NIP. 19781201200212 2002**

SURABAYA
2 JULI 2015

DESAIN KONTROLER *FUZZY* PID *GAIN SCHEDULING* UNTUK PENGATURAN KECEPATAN MOTOR DC TANPA SIKAT

Hudaibiy Hibban
2211 100 108

Dosen Pembimbing I : Ir. Josaphat Pramudijanto, M.Eng.
Dosen Pembimbing II : Nurlita Gamayanti ST., MT.

ABSTRAK

Motor DC, sebagai salah satu mesin listrik yang banyak digunakan di industri, penggunaannya banyak digantikan dengan *Brushless DC Motor* (BLDC). Penggunaan motor jenis ini hampir tidak menimbulkan polusi di jalan karena energi yang dipergunakan adalah energi listrik, sehingga selain kenyamanan karena nyaris tidak menimbulkan kebisingan motor ini juga ramah lingkungan. Di dalam tugas akhir ini, saya merancang sebuah sistem yang memberikan pembebanan pada motor BLDC menggunakan rem magnetik agar sistem menjadi tidak linear. Pada penelitian ini, digunakan kontroler *Fuzzy* PID *Gain Scheduling* untuk mengatur kecapatan motor BLDC. Dipilih kontroler PID karena pada kenyataannya kontrol yang paling sering dipakai adalah PID, dan *Fuzzy* digunakan sebagai *tuning* nilai K_p, K_i, dan K_d. Setelah dilakukan implementasi pada kontroler didapatkan bahwa nilai *rise time* rata-rata respon motor sebesar 2,6 detik dan nilai *settling time* rata-rata sebesar 3,6 detik. Selain itu terdapat juga overshoot yang relatif kecil sekitar 1 persen. Hasil tugas akhir berupa motor BLDC dapat digunakan sebagai penelitian lebih lanjut untuk adik-adik kelas di Jurusan Teknik Elektro maupun jurusan lain.

Kata Kunci : *Brushless DC, Fuzzy PID Gain Scheduling*

DESIGN FUZZY PID GAIN SCHEDULING CONTROLLER FOR BRUSHLESS DC MOTOR SPEED CONTROL

Hudaibiy Hibban
2211 100 108

Supervisor I : Ir. Josaphat Pramudijanto, M.Eng.
Supervisor II : Nurlita Gamayanti ST., MT.

ABSTRACT

DC motors, as one of the electrical machines are widely used in industrial, are now replaced with Brushless DC Motors (BLDC). The application of this type of motor will not cause any pollution on the road because motor BLDC consumes electrical energy, so motor BLDC is not only good because it almost not cause any commotion but also environmentally friendly. In this final project, I design a system that will give a brake to BLDC motor using a magnetic brake so the system will become nonlinear. In this study, PID controller Fuzzy Gain Scheduling are used to control speed of BLDC motor. PID controller are selected because in fact the most frequently used controls are PID and Fuzzy is used to tune the value of K_p, K_i, and K_d. After this controller implented on motor BLDC we got that this speed respon have around 2,6 seconds of rise teime and aroud 3,6 seconds of settling time. This respon also have an overshoot about 1 percent. If this final project is successful .this BLDC motors can be used as further research for juniors in the department of Electrical and other majors.

Keywords **:** *Brushless DC, Fuzzy PID Gain Scheduling*

KATA PENGANTAR

Alhamdullilah, puji syukur penulis panjatkan kehadirat Allah SWT karena atas rahmat dan karunia-Nya penulis dapat menyelesaikan penulisan buku tugas akhir dengan judul **"DESAIN KONTROLER *FUZZY* PID *GAIN SCHEDULING* UNTUK PENGATURAN KECEPATAN MOTOR DC TANPA SIKAT"**. Tugas akhir merupakan salah satu syarat yang harus dipenuhi untuk menyelesaikan program studi Strata-1 pada Jurusan Teknik Elektro Fakultas Teknologi Industri Institut Teknologi Sepuluh Nopember Surabaya.

Penulis menyadari bahwa dalam penulisan tugas akhir ini banyak mengalami kendala, namun berkat bantuan, bimbingan, dan kerja sama dari berbagai pihak sehingga kendala-kendala tersebut dapat diatasi. Untuk itu pada kesempatan ini penulis menyampaikan banyak terimakasih dan penghargaan setingi-tingginya kepada :

1. Kedua orang tua, Syaiful Anshori dan Ibunda Fauzia Tanjung yang selalu memberikan dukungan, semangat, dan doa kepada penulis.
2. Bapak Josaphat dan Ibu Nurlita selaku Dosen Pembimbing atas segala bantuan, perhatian, dan arahan selama pengerjaan tugas akhir ini.
3. Bapak Rusdi selaku Koordinator Bidang Studi Sistem Pengaturan Jurusan Teknik Elektro ITS.
4. Bapak Tri Arief Sardjono selaku Ketua Jurusan Teknik Elektro ITS.
5. Rekan-rekan e51 khususnya bidang studi Sistem Pengaturan.
6. Mas Fahrul mahasiswa S2 Teknik Elektro sebagai pembimbing kami dalam merancang dan membuat alat.
7. Teman-teman seperjuangan, dan teman satu kontrakan.

Penulis berharap tugas akhir ini dapat bermanfaat bagi yang membutuhkannya.

Surabaya, 2 Juli 2015

Penulis

DAFTAR ISI

Halaman ini sengaja dikosongkan

DAFTAR GAMBAR

DAFTAR TABEL

Halaman ini sengaja dikosongkan

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Penggunaan motor listrik dalam aplikasi sehari-hari semakin meningkat. Motor listrik banyak digunakan dalam berbagai peralatan seperti: *air conditioning, vacuum cleaner, conveyor*, lemari pendingin, dan lain sebagainya. Dan yang terutama motor listrik ini juga mulai dipakai pada aplikasi mobil listrik. Motor *universal* dan motor *direct current* (DC) banyak digunakan dalam aplikasi tersebut. Akan tetapi penggunaan motor DC konvensional menimbulkan masalah diakibatkan oleh penggunaan sikat. Penggantian sikat secara periodik untuk menjaga kinerja serta busur api adalah masalah yang umum disoroti pada motor DC. Oleh karena itu pada tugas akhir ini dipilihlah *plant* motor *Brushless Direct Current* (BLDC) yang merupakan alternatif pengganti motor DC. Motor ini adalah salah satu jenis motor yang popularitasnya mulai naik. [1]

Pada tugas akhir ini digunakan metode *Fuzzy* PID *gain scheduling* untuk mengendalikan kecepatan motor BLDC. Metode ini dipilih karena pada aplikasinya di dunia industri kebanyak kontroler yang dipakai adalah kontroler PID. Dan *Fuzzy* di sini berfungsi sebagai pengatur nilai Kp, Ki, dan Kd pada PID karena pada implementasinya digunakan pembebanan yang berubah-ubah.

1.2 Perumusan Masalah

Permasalahan di sini adalah pada saat keadaan nyata jalanan tidak selalu datar, banyak terdapat jalanan menanjak sehingga yang menjadi masalah pada tugas akhir ini adalah bagaimana mengatur Torsi motor BLDC agar dapat menanggung atau mengatasi beban yang berubah-ubah secara tidak menentu. Oleh karena itu, dibutuhkan suatu kontroler untuk melakukan pengaturan kerja dari motor listrik agar bekerja sesuai dengan kebutuhan yang diinginkan.

1.3 Batasan Masalah

Permasalahan pada tugas akhir ini dibatasi oleh beberapa hal antara lain:

a. Kecepatan pada poros yang ada pada *plant* berkisar antara 300 rpm hingga 3500 rpm. Pada tugas akhir ini kecepatan yang dikontrol dimulai dari 500 hingga 1750 rpm

b. Kecepatan putar BLDC merupakan kecepatan yang terukur pada poros beban, sehingga kecepatan terukur sebesar sekitar 25% kecepatan aktual

1.4 Tujuan Penelitian

Tujuan dari pelaksanaan tugas akhir ini adalah:

a. Membuat *plant* berupa motor BLDC yang terangkai dengan rem magnetik yang berfungsi sebagai pembebanan pada motor BLDC.
b. Merancang kontroler *Fuzzy PID Gain Scheduling* untuk mengatasi permasalahan adanya efek pembebanan yang berubah-ubah pada motor BLDC.
c. Implementasi pada Motor BLDC yang telah dirancang menggunakan *software* MATLAB dan mikrokontroler Arduino dengan harapan dapat memperbaiki performansi kerja pada motor BLDC dan mendapatkan performansi terbaik untuk pengendalian torsi motor BLDC pada pembebanan yang berbeda-beda.

Hasil yang diperoleh dari pelaksanaan tugas akhir ini diharapkan dapat memberikan manfaat dan kontribusi bagi dunia pendidikan, industri dan masyarakat agar dapat dijadikan referensi bagi peneliti lainnya dan sebagai ilmu pengetahuan.

1.5 Sistematika Penulisan

Buku tugas akhir ini terdiri dari lima bab dan disusun menurut sistematika penulisan berikut ini

BAB I: PENDAHULUAN

Bab ini berisi tentang latar belakang, rumusan masalah, tujuan, batasan masalah, sistematika penulisan, dan relevansi.

BAB 2: DASAR TEORI

Bab ini berisi tentang teori yang menunjang penelitian, berupa teori tentang BLDC dan komponennya, serta metode yang digunakan untuk pengaturan kecepatan motor BLDC.

BAB 3: PERANCANGAN SISTEM DAN KONTROLER

Bab ini berisi tentang perancangan perangkat keras, perangkat lunak, dan perancangan kontroler.

BAB 4: PENGUJIAN DAN ANALISA

Bab ini berisi tentang hasil simulasi kontroler dan analisanya. Selain itu berisi tentang hasil implementasi kontroler pada Simulator BLDC beserta analisa hasil implementasi.

BAB 5: KESIMPULAN DAN SARAN

Berisi kesimpulan dan saran yang dapat dijadikan pertimbangan pengembangan berdasar hasil pengerjaan tugas akhir ini.

1.6 Relevansi

Hasil dari tugas akhir ini diharapkan dapat memberikan manfaat dalam pengembangan penelitian tentang BLDC khususnya strategi kontrol motor listrik dalam BLDC.

Halaman ini sengaja dikosongkan

BAB 2
TINJAUAN PUSTAKA

2.1 Motor *Brushless* DC

BLDC motor atau dapat disebut juga dengan BLAC motor merupakan motor listrik *synchronous* AC 3 fasa. Perbedaan pemberian nama ini terjadi karena BLDC memiliki BEMF berbentuk *trapezoid*, sedangkan BLAC memiliki BEMF berbentuk sinusoidal. Walaupun demikian keduanya memiliki struktur yang sama dan dapat dikendalikan dengan metode *six-step* maupun PWM sinusoidal. Dibandingkan dengan motor DC, BLDC memiliki biaya perawatan yang lebih rendah dan kecepatan yang lebih tinggi akibat tidak digunakannya *brush*. Dibandingkan dengan motor induksi, BLCM memiliki efisiensi yang lebih tinggi karena rotor dan torsi awal yang lebih tinggi karena rotor terbuat dari magnet permanen. Walaupun memiliki kelebihan dibandingkan dengan motor DC dan motor induksi, pengendalian BLDC jauh lebih rumit untuk kecepatan dan torsi yang konstan karena tidak adanya *brush* yang menunjang proses komutasi dan harga BLDC jauh lebih mahal. Secara umum BLDC terdiri dari dua bagian, yakni rotor, bagian yang bergerak, yang terbuat dari permanen magnet dan stator, bagian yang tidak bergerak, yang terbuat dari kumparan 3 fasa. Walaupun merupakan motor listrik *synchronous* AC 3 fasa, motor ini tetap disebut dengan BLDC karena pada implementasinya BLDC menggunakan sumber DC sebagai sumber energi utama yang kemudian diubah menjadi tegangan AC dengan menggunakan *inverter* 3 fasa.

Tujuan dari pemberian tegangan AC 3 fasa pada stator BLDC adalah menciptakan medan magnet putar stator untuk menarik magnet rotor. Oleh karena tidak adanya *brush* pada motor BLDC, untuk menentukan *timing* komutasi yang tepat pada motor ini sehingga didapatkan torsi dan kecepatan yang konstan, diperlukan 3 buah sensor h*all* dan atau *encoder*. Pada sensor h*all*, *timing* komutasi ditentukan dengan cara mendeteksi medan magnet rotor dengan menggunakan 3 buah sensor *hall* untuk mendapatkan 6 kombinasi *timing* yang berbeda, sedangkan pada *encoder*, *timing* komutasi ditentukan dengan cara menghitung jumlah pola yang ada pada *encoder*. Pada umumnya *encoder* lebih banyak digunakan pada motor BLDC komersial karena *encoder* cenderung mampu menentukan *timing* komutasi lebih presisi

dibandingkan dengan menggunakan sensor hall. Hal ini terjadi karena pada *encoder*, kode komutasi telah ditetapkan secara *fixed* berdasarkan banyak kutub dari motor dan kode inilah yang digunakan untuk menentukan *timing* komutasi. Namun karena kode komutasi *encoder* untuk suatu motor tidak dapat digunakan untuk motor dengan jumlah kutub yang berbeda. Hal ini berbeda dengan sensor *hall*. Apabila terjadi perubahan pole rotor pada motor, posisi sensor *hall* dapat diubah dengan mudah. Hanya saja kelemahan dari sensor *hall* adalah apabila posisi sensor *hall* tidak tepat akan terjadi keselahan dalam penentuan *timing* komutasi atau bahkan tidak didapatkan 6 kombinasi *timing* komutasi yang berbeda. [2]

Pada tugas akhir kali ini motor yang kami gunakan adalah motor yang dipakai pada *quadcopter*. Dan untuk lebih jelasnya bentuk dari motor ini dapat dilihat pada Gambar 2.1

Gambar 2.1 BLDC.

2.1.1 Cara Kerja Motor *Brushless* DC

Cara kerja pada motor BLDC cukup sederhana, yaitu magnet yang berada pada poros motor akan tertarik dan terdorong oleh gaya elektromagnetik yang diatur oleh *Electronic Speed Controller* (ESC). Hal ini membedakakn motor BLDC dengan motor DC yang

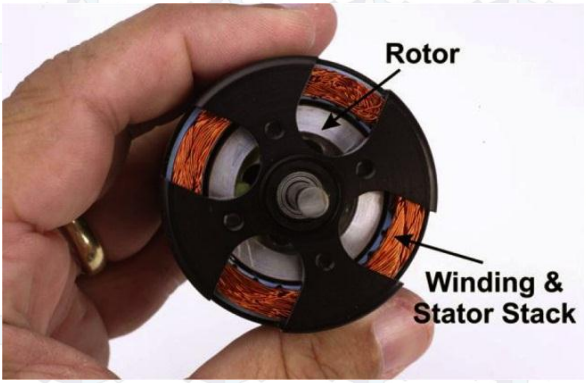
menggunakan sikat mekanis yang berada pada komutator untuk mengatur waktu dan memberikan medan magnet pada lilitan. Motor BLDC ini juga berbeda dengan motor AC yang pada umumnya menggunakan siklus tenaga sendiri untuk mengatur waktu dan memberi daya pada lilitan. BLDC dapat memberikan rasio daya dan beban yang lebih tinggi secara signifikan dan memberikan efisiensi yang lebih baik dibandingkan motor tanpa sikat tradisional. [3]

2.1.2 Konstruksi Motor *Brushless* DC

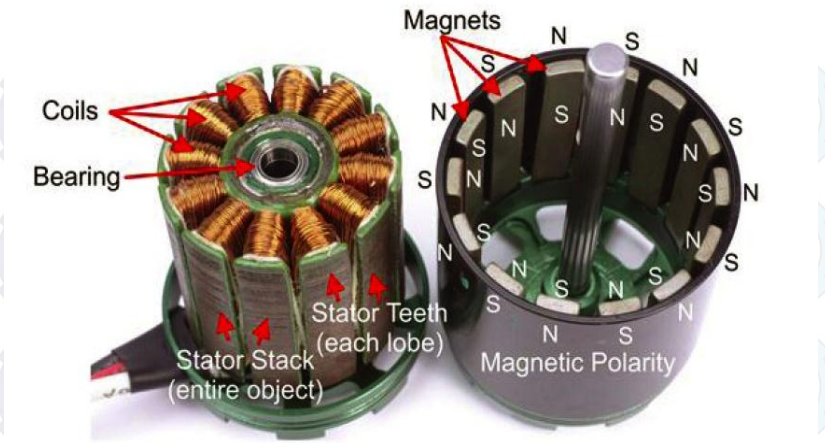
Motor BLDC terdiri dari 2 jenis yaitu *outrunner* dan *inrunner*. *Outrunner* merupakan jenis motor BLDC yang memiliki stator terletak di dalam motor dan rotor terletak di luar sehingga bagian dari motor yang berputar merupakan bagian luar dari motor. Sedangkan *inrunner* merupakan jenis motor yang memiliki stator di bagian luar motor dan rotor berada pada bagian dalam motor sehingga bagian motor yang berputar adalah bagian dalam motor.

Stator merupakan bagian motor yang memiliki lilitan, sehingga apabila pada lilitan rotor diberi arus akan timbul gaya elektromagnetik. Dan stator merupakan bagian motor yang diberi magnet permanen. Oleh karena itu, apabila pada rotor diberi arus maka pada pada stator dan rotor akan timbul gaya tarik-menarik dan tolak-menolak yang akan menyebabkan motor akan berputar.

BLDC yang paling sering digunakan pada RC (*Remote Control*) pesawat dan helikopter adalah motor jenis *outrunner*.



Gambar 2.2 *Inrunner* BLDC. [3]



Gambar 2.3 Konstruksi Motor BLDC *Outrunner* [3]

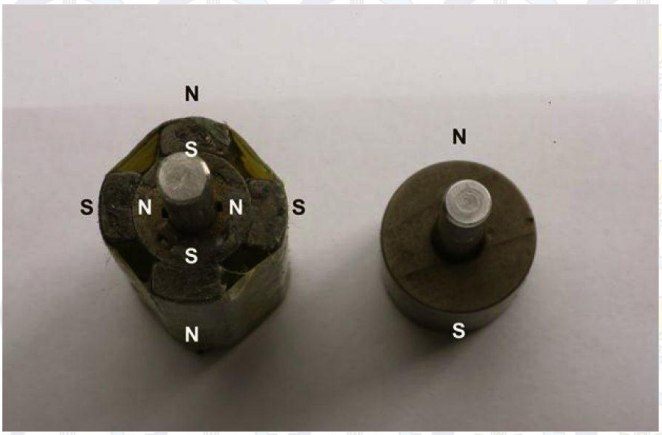
2.1.3 Magnet Permanen

Magnet permanen pada motor BLDC diatur sedemikian rupa sehingga kutubnya tegak lurus dengan poros dari motor. Mereka disusun sedemikian rupa sehingga setiap kutub yang dihadapkan dengan lilitan elektromagnetik hanya merupakan kutub utara atau selatan. Magnet permanen ini disusun dengan kutub yang berubah-ubah di sekelilingnya sehingga apabila terdapat kutub yang mendekati, kutub ini akan ditarik dan didorong oleh medan magnet yang sama. Motor 2 kutub bisa terbuat dai sebuah magnet yang dilingkarkan pada poros sehingga pada magnet terdapat kutub utara disalah satu sisinya dan kutub selatan di sisi lainnya. [3]

Motor *outrunner* menggunakan lilitan elektromagnetik di bagian tengah pada motor untuk memutar magnet permanen yang terdapat di luarnya. Desain ini dapat memberikan diameter rotor yang lebih besar dan bisa meberikan torsi yang lebih besar dibandingkan dengan motor *inrunner* yang berdiameter sama.

Motor *inrunner* biasanya memiliki magnet permanen yang menempel pada poros, seperti pada Gambar 2.4 di bawah. Rotor 2 kutub memiliki magnet yang mengelilingi poros. Integritas dari pemasangan magnet permanen ini mempengaruhi kecepatan maksimal dari sebuah motor. Beberapa motor seperti motor yang berada di kanan pada

Gambar 2.4 menggunakan *kevlar* atau karbon untuk memperkuat pemasangannya pada rotor.

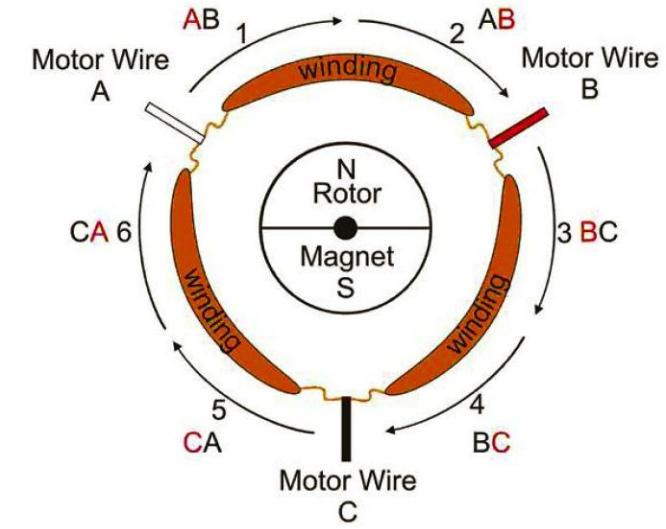


Gambar 2.4 Rotor *Inrunner* 4 Kutub dan 2 Kutub [3]

2.1.4 *Wiring* Motor BLDC

Semua motor BLDC dari tipe Hobby memiliki 3 kabel yang terhubung dengan lilitan motor dan misalnya saja lilitan ‘A’, ‘B’, dan ‘C’. Pada bagian akhir dari lilitan terhubung dengan kontroler dan bagian akhir dari lilitan lain bisa terhubung atau terputus, tergantung tipe *winding* yang dipakai yaitu tipe DELTA atau WYE. Walaiupun kedua tipe ini terlihat berbada, hal ini tidak akan merubah tugas ESC yang menjalankan motor. [3]

Masukkan sebuah sumber pada salah satu belitan, atau fasa, dengan menghubungkan kawat pada positif dan yang lain pada negatif dan hal ini akan menyebabkan medan magnetyang menarik dan mendorong magnet permanen. Lalu apabila polaritas dari arus dibalik maka efek medan magnet juga akan berbalik. Hal ini berfungsi untuk menarik dan medorong magnet kutub pada rotor yang bergerak. Ada enam kombinasi kutub yang akan terjadi pada kutub elektromagnetik dan semua enam kombinasi ini akan digunakan untuk menjalankan satu magnet kutub dalam satu putaran pada diagram motor yang telah disederhanakan di Gambar 2.5.



Gambar 2.5 *Simplified Motor Diagram* [3]

Sekuen yang tepat pada pengaplikasian listrik dan sebaliknya dibutuhkan untuk memutar poros dalam satu putaran. Tiga belitan pada Gambar 2.5 di atas diberi energi yang berkebalikan menggunakan 3 masukan motor, A, B, dan C. Kombinasinya terlihat pada Gambar 2.5 yaitu apabila huruf berwana merah berarti huruf tersebut diberi polaritas yang positif dan untuk huruf berwarna hitam memiliki polaritas yang negatif. Pada gambar tersebut diperliahatkan motor 2 kutub yang setiap fasanya hanya memiliki satu belitan.

Desain dan kualitas material dari komponen motor memiliki efek langsung ke performa motor. Faktor desain termasuk:

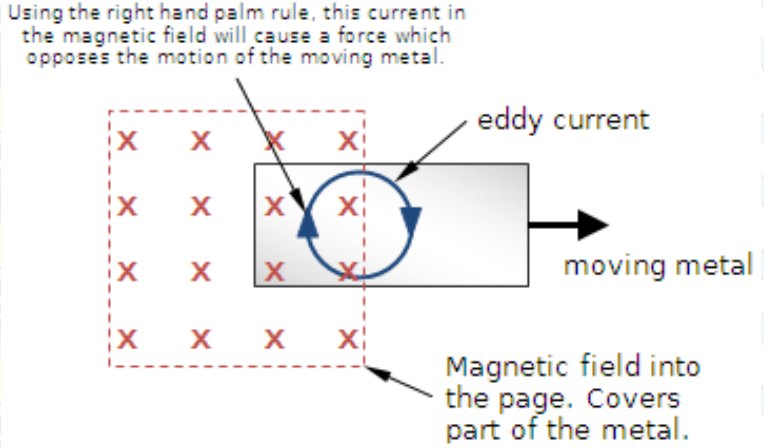
a. banyaknya tembaga yang ada pada motor.
b. celah udara, atau jarak antara magnet rotor dan belitan.
c. material, ketebalan dan bentuk dari baja yang digunakan untuk laminasi pada stator.
d. bahan dan kekuatan dari magnet permanen.
e. *bearing* juga penting, begitu juga dengan kesimbangan rotor.

2.1.5 ESC (*Electronic Speed Controller*)

Penjelasan yang cepat mengenai ESC adalah ESC harus secara akurat menghubungkan dan memutuskan koneksi antara 3 masukan *input* dan 3 belitan pada stator agar rotor dapat berputar. [3]

2.2 Rem Elektromagnetik

Sistem pengereman ini menggunakan gaya elektromagnetik yang timbul dari suatu magnet permanen tetap atau kumparan yang diberikan arus listrik untuk memperlambat suatu gerakan. Konstruksi dasarnya berupa suatu piringan logam non-feromagnetik yang terpasang pada suatu poros yang berputar. Piringan logam tersebut diapit oleh kumparan yang dialiri arus listrik hingga menimbulkan medan magnet yang kutubnya saling berlawanan. Logam piringan tersebut akan memotong medan magnet yang ditimbulkan oleh kumparan tersebut sehingga menimbulkan *eddy current* atau arus *eddy*.

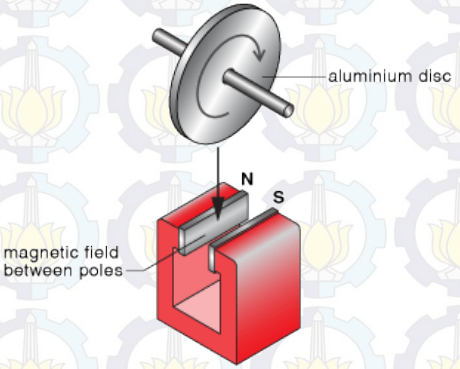


Gambar 2.6 Prinsip Arus *Eddy* Pada Logam yang Bergerak [4]

Arus *eddy* merupakan arus listrik yang timbul bilamana suatu piringan logam berada di sekitar medan magnet yang garis-garis gayanya sedang berubah-ubah. Arus *eddy* ini mempunyai medan magnet yang arahnya berlawanan dengan arah gerak piringan logam. Akibatnya laju piringan logam akan tertahan akibat dari adanya arus *eddy* ini.

Rem elektromagnetik biasa diletakkan dekat dengan bagian yang bergerak. Rem ini bekerja pada kondisi yang dingin dan memenuhi

persyaratan energi pengereman kecepatan tinggi karena tanpa adanya gesekan. Selain pada kendaraan listrik, aplikasi rem elektromagnetik juga dapat ditemukan pada sistem pengereman kereta api. Gambar 2.7 akan memperlihatkan struktur dan konstruksi dari sebuah sistem rem elektromagnetik.



Gambar 2.7 Struktur dari Sebuah Rem Elektromagnetik. [5]

2.3 Sensor *Rotary Encoder*

Rotary Encoder adalah suatu komponen elektro mekanis yang memiliki fungsi untuk memonitoring posisi anguler pada suatu poros yang berputar. Dari perputaran benda tersebut data yang terpantau akan diubah ke dalam bentuk data digital oleh *rotary encoder* berupa lebar pulsa kemudian akan dihubungkan ke kontroler (Mikrokontroler/PLC). Berdasarkan data yang di dapat berupa posisi anguler (sudut) kemudian dapat diolah oleh kontroler sehingga mendapatkan data berupa kecepatan, arah, dan posisi dari perputaran porosnya.

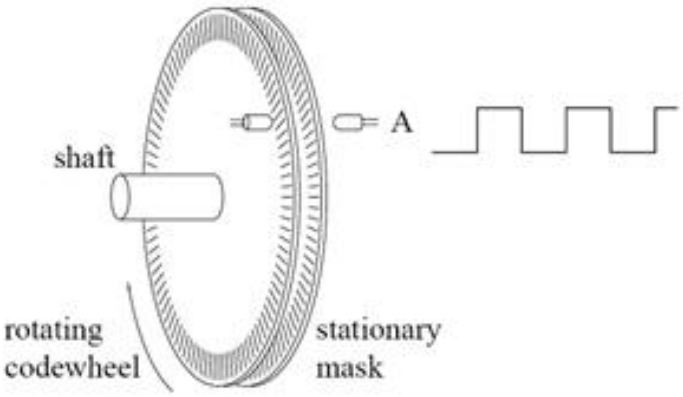
Penerapan dari penggunaan *rotary encoder* sering dijumpai pada robot-robot yang membutuhkan kepresisian tinggi dalam hal posisi seperti Robot *Mechanum* dan Robot Omni, selain itu untuk robot berjenis *Differential Drive* (ex : Robot Tank) juga disarankan menggunakan *rotary encoder* untuk mengatur agar kecepatan putar motor di roda kiri dan kanan bisa sama.

Konstruksi *rotary encoder* berupa piringan tipis yang biasanya dikopel dengan poros yang berputar, umumnya dikopel langsung dengan shaft motor. Piringan tipis tersebut terdapat lubang di sepanjang pinggir lingkarannya. Di bagian sisi-sisi piringan terdapat sebuah LED dan

phototransistor di bagian bersebrangan. Fungsi dari lubang-lubang yang berada di sepanjang pinggir lingkaran tersebut akan menghantarkan cahaya LED ke *phototransistor*, sebaliknya jika cahaya led tidak menembus lubang piringan maka cahaya akan tertahan. Piringan tersebut akan berputar sesuai dengan kecepatan putaran motor sehingga phototransistor akan saturasi ketika cahaya led menembus lubang-lubangnya.

Pada saat saturasi *phototransistor* akan menghasilkan pulsa dengan range 0,5V s/d 5V. Semakin banyak lubang yang berada pada piringan tentu saja semakin banyak pulsa yang dihasilkan selama satu putaran, hal tersebut berbanding lurus dengan tingkat akurasi yang dihasilkan oleh *rotary encoder*. [6]

Pada saat *rotary encoder* berputar maka phototransistor akan mengirimkan pulsa-pulsa yang nantinya akan dimasukkan ke kontroler. Data pulsa itu nantinya akan di ubah dengan mengitung kecepatan atau waktu setiap pulsanya atau frekuensi pulsanya. Data frekuensi akan diubah menjadi data berupa satuan rpm.



Gambar 2.8 *Rotary Encoder.* [6]

2.4 Arduino Uno [7]

Arduino Uno adalah *board* mikrokontroler berbasis Atmega328. Alat ini mempunyai 14 pin *input*/*output* digital dan 6 diantaranya dapat digunakan sebagai *output* PWM, 6 *input* analog, resonator keramik

19MHz, koneksi USB, soket listrik, ICSP *header*, dan tombol reset. Dengan kabel USB alat ini mudah dihubungkan ke komputer dan dapat diaktifkan dengan baterai atau adaptor AC ke DC. Spesifikasi Arduino Uno adalah sebagai berikut :

a. Mikrokontroler : Atmega328
b. Tegangan operasi : 5V
c. Tegangan *input* (direkomendasikan) : 7-12V
d. Tegangan *input* (batasan) : 6-20V
e. Pin *input*/*output* digital : 14 (6 diantaranya *output* PWM)
f. Pin *input* analog : 6
g. Arus DC tiap pin I/O : 40mA
h. Arus DC untuk pin 3,3 V : 50mA

14 pin *input* dan *output* Arduino Uno dapat digunakan dengan fungsi *pinMode(), digitalWrite(),* dan *digitalRead*(). Tiap pin memiliki tegangan operasi 5V dan dapat menerima arus maksimum 40mA. Beberapa pin memiliki fungsi khusus

a. *Serial* : 0 (RX) dan 1 (TX). Digunakan untuk menerima (RX) dan mengirim (TX) data serial TTL. Pin ini terhubung dengan pin pada Atmega8U2 USB ke cip USB ke TTL serial
b. *External Interrupts* : 2 dan 3.pin ini digunakan sebagai *trigger* gangguan pada nilai yang rendah, menaikkan dan menurunkan nilai, atau merubah nilai.
c. PWM : 3,5,6,9,10, dan 11. Menyediakan *output* PWM 8bit dengan fungsi *analogWrite().*
d. SPI (*Serial Peripheral Interface*) : 10 (*Slave Select*), 11 (*Master Out Slave In*), 12 (*Master In Slave Out*), 13 (*Serial Clock*). Pin ini mendukung komunikasi SPI.
e. LED : 13, LED ini terhubung dengan pin 13. Ketika pin memiliki nilai yang tinggi LED akan *on* dan ketika pin memiliki nilai yang rendah, LED akan *off*

Pada Arduino Uno terdapat 6 *input* analog, dengan nama A0 hingga A5. Pin tersebut memiliki resolusi 10-bit (1024 nilai yang berbeda). *Input* analog ini berupa tegangan dari *ground* ke 5V.

2.5 MATLAB [7]

MATLAB merupakan paket program dengan bahasa pemrograman yang tinggi untuk mengembangkan algoritma, visualisasi data, dan komputasi numerik. Program MATLAB ini dapat digunakan untuk menyelesaikan masalah komputasi dengan lebih cepat dibandingkan dengan bahasa pemrograman tradisional, seperti C, C++, dan Fortran. MATLAB digunakan untuk banyak aplikasi seperti *signal and image processing*, desain kontrol, pengujian dan pengukuran, permodelan, dan analisis.

Simulink merupakan bagian dari MATLAB untuk memodelkan, mensimulasikan, dan menganalisa sistem dinamik. Simulink dapat membentuk model dari awal atau memodifikasi model yang sudah ada sesuai dengan apa yang diinginkan. Selain itu simulink juga mendukung sistem *linier* dan *non-linier*, permodelan waktu kontinyu atau diskrit, atau gabungan. Simulink ini dapat digunakan sebagai media untuk menyelesaikan masalah dalam industri nyata meliputi kedirgantaraan dan pertahanan, otomotif, komunikasi, elektronik dan pemrosesan sinyal,

Salah satu modul dalam Simulink yang dapat digunakan untuk komunikasi perangkat keras adalah *Instrument Control Toolbox*. Modul ini merupakan kunpulan fungsi *m-file* yang dibangun pada lingkungan komputasi teknis MATLAB. *Toolbox* ini menyediakan kerangka kerja untuk komunikasi instrumen yang mendukung GPIB *interface*, standar VISA, TCP/IP, dan protokol UDP. *Toolbox* ini memperluas fitur dasar *serial port* yang ada dalam MATLAB. Selain itu *toolbox* ini berfugsi untuk komunikasi data antara *workspace* MATLAB dan peralatan lainnya. Data tersebut dapat berbentuk biner atau *text*.

Komunikasi *serial* merupakan protokol dasar tingkat rendah untuk komunikasi antara dua peralatan atau lebih. Pada umumnya satu komputer dengan modem, *printer*, mikrokontroler, atau peralatan lainnya. *Serial port* mengirim dan menerima informasi *bytes* dengan hubungan seri. *bytes* tersebut dikirimkan menggunakan format biner atau karakter ASCII (*American Standard Code for Information Interchange*). Dalam komunikasi serial MATLAB, agar data ASCII dapat diproses *real time*, maka digunakan ASCII *encode* dan *decode* yang terdapat pada *xPC Target Library for RS232*. ASCII *encode* merupakan blok dalam simulink yang digunakan untuk mengubah data *bytes* menjadi karakter ASCII. Sedangkan ASCII *decode* merupakan

blok Simulink yang digunakan untuk mengubah karakter ASCII menjadi data *bytes* yang kemudian dapat dikonversi sesuai kebutuhan.

2.6 Identifikasi Sistem [8]

Identifikasi sistem merupakan suatu langkah awal dalam menganalisa sistem dinamik. Menurunkan suatu model matematika yang baik dan sesuai merupakan salah satu bagian terpenting dalam proses menganalisa sebuah sistem secara keseluruhan. Model matematika yang baik dan sesuai cocok digunakan untuk analisa, prediksi, dan desain sistem, *regulator*, dan *filter*. Model matematika memiliki bentuk yang bermacam-macam. Salah satu bentuknya adalah *transfer function* yang cocok untuk permasalahan analisa transien dan sebuah sistem LTI (*Linear Time Invariant*) dan sistem dengan *Single-Input Single-Output* (SISO). Disisi lain, model matematika dengan bentuk *state space* sangat cocok untuk menganalisa suatu sistem dengan *Multiple-Input Multiple-Output* (MIMO).

Model matematika dari sebuah sistem dapat diperoleh melalui dua cara yaitu permodelan fisik dan identifikasi sistem. Permodelan fisik merupakan pendekatan analitik berdasarkan hukum fisika seperti hukum newton dan hukum kesetimbangan. Permodelan ini menjelaskan dinamika dalam sistem. Yang kedua adalah identifikasi sistem. Hal ini dilakukan dengan pendekatan eksperimental. Model ini berdasarkan data dari eksperimen yang kemudian didapatkan nilai parameter sistem. Pada beberapa kasus yang sangat kompleks, sangat sulit untuk menentukan model berdasarkan pemahaman fisik.

Identifikasi sistem pada suatu *plant* dapat dilakukan dengan menggunakan metode identifikasi statis maupun dinamis. Identifikasi statis dilakukan untuk mendapatkan *gain* dan *time sampling* dari suatu *plant*. Sedangkan identifikasi dinamis digunakan untuk mendapatkan model matematika dari suatu *plant* yang menggambarkan hubungan antara *input* dan *output* pada kondisi berbeban. Pada tugas akhir kali ini, digunakan identifikasi dinamis dengan menggunakan model pendekatan ARX.

2.6.1 Identifikasi Sistem ARX

Metode Sistem dinamik secara umum dapat dimodelkan dengan menggunakan persamaan deret domain waktu seperti pada persamaan 2.1, yang menyatakan relasi antara variabel *input* u(t) *exogenous* (ekstra atau yang diketahui) dan *output* y(t)

$$A(q)y(t) = \sum_{i=1}^{nu} B_i\,(q)u_i(t - nk_i) + e(t) \quad (2.1)$$

dengan e(t) merupakan nilai rerata gangguan (*noise*) yang diasumsikan sebagai "*white Gausian noise*". Parameter sistem atau koefisien A, B dan C dapat ditentukan dengan proses perhitungan regresi berdasarkan rata-rata bergerak kuadrat kesalahan terkecil. [9]

2.6.2 *Root Mean Square Error*

Persamaan permodelan digunakan pada banyak bidang ilmu, di mana variabelnya harus diukur dengan ada tidaknya *error*. Tujuannya adalah agar nilai pada permodelan mendekati nilai *real* sistem.

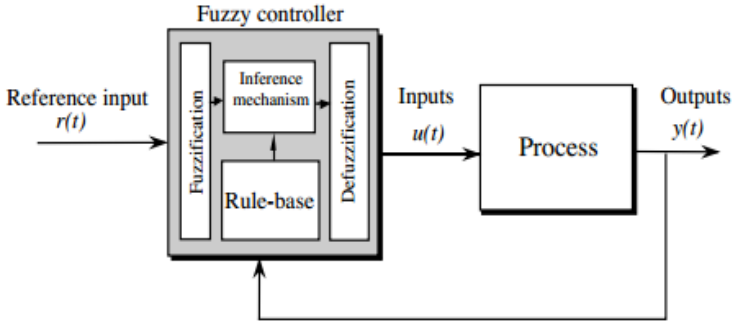
Untuk melakukan validasi model dapat digunakan beberapa tolak ukur. Salah satunya adalah RMSE. Dalam penggunaanya, harus ditentukan terlebih dahulu jumlah data dan interval toleransi RMSE. Jika nilai RMSE kecil, menunjukkan bahwa nilai parameter sesuai dengan apa yang diinginkan sebelumnya. [7]

Dengan e adalah *error* antara nilai estimasi dan nilai sebenarnya dan n adalah jumlah data.

$$RMSE = \sqrt{\frac{\sum e^2}{n}} \quad (2.2)$$

2.7 *Fuzzy Logic*

Logika *fuzzy* pertama kali diperkenalkan oleh Prof Lotfi Zadeh tahun 1965. Saat ini logika *fuzzy* sudah banyak digunakan dalam berbagai bidang, salah satunya adalah dalam bidang kontrol (proses kontrol). *fuzzy* yang diperkenalkan oleh Zadeh didasarkan pada teori possibilitas yang berbeda dari teori probabilitas yang sudah lebih umum dikenal. Secara umum, logika *fuzzy* terdiri dari beberapa komponen, yaitu *fuzzifier*, *fuzzy rule base*, *fuzzy inference mechanisme* atau *fuzzy inference engine*, dan *defuzzifier*, seperti diperlihatkan pada gambar berikut. [10]



Gambar 2.9 *Fuzzy Controller* [11].

Yang menjadi inti dari logika *fuzzy* adalah *fuzzy rule base*, yang berisi pernyataan-pernyataan logika. *Fuzzy inference engine* merupakan komponen *fuzzy* yang menerjemahkan pernyataan logika yang ada di *Rule Base* menjadi perhitungan-perhitungan matematika. *Fuzzifier* digunakan untuk memetakan nilai/harga variable di dunia nyata kedalam himpunan *fuzzy* (*fuzzy sets*), sedangkan *defuzzifier* mengembalikan hasil perhitungan *fuzzy* (himpunan *fuzzy*) menjadi variabel sesuai rentang nilainya di dunia nyata.

2.7.1 *Fuzzy Rule Base*

Fuzzy rule base berisi pernyataan-pernyataan logika *fuzzy* (*fuzzy statement*), yang berbentuk pernyataan IF-THEN. Bentuk umum pernyataan *fuzzy* adalah:

$$\text{IF } x_1 \text{ is } A_1^l \text{ and } \ldots \text{ and } x_n \text{ is } A_n^l \text{ THEN } y \text{ is } B^l \qquad (2.3)$$

A_1^l dan B^l adalah himpunan *fuzzy*, sedangkan $x = (x_1, x_2, \ldots, x_n)^T$ dan y adalah *input* dan *output* dari *variable fuzzy*. [10]

2.7.2 *Fuzzy Interference Engine*

Fuzzy inference engine menerjemahkan pernyataan-pernyataan *fuzzy* dalalm *rule base* menjadi perhitungan matematika (*fuzzy combinational*) [10]. Terdapat beberapa metode *inference engine*, 5 diantaranya yaitu:

a. *Product Inference Engine.*
b. *Minimum Inference Engine.*

c. *Lukasiewicz Inference Engine.*
d. *Zadeh Inference Engine*.
e. *Dienes-Rescher Inference Engine.*

2.7.3 Fuzzification

Fuzzifier digunakan untuk memetakan nilai/harga variabel di dunia nyata kedalam himpunan *fuzzy* (*fuzzy sets*). Pemetaannya dilakukan dengan menggunakan fungsi yang disebut fungsi keanggotaan. Terdapat beberapa metode *fuzzifier*, 3 diantaranya yaitu *singleton fuzzifier*, *gausian fuzzifier* dan *triangular fuzzifier*. Berikut adalah formulanya.

a. *Singleton fuzzyfier*

$$\mu A'(x) = \begin{cases} 1 \; if \; x = x* \\ 0 \; otherwise \end{cases} \tag{2.4}$$

b. *Gaussian fuzzifier*

$$\mu A'(x) = e^{-\left(\frac{x_1 - x_1^*}{a_1}\right)} * \ldots . * e^{-\left(\frac{x_n - x_n^*}{a_n}\right)} \tag{2.5}$$

c. *Triangular fuzzyfier*

$$\mu A'(x) = \begin{cases} \left(1 - e^{-\left(\frac{x_1 - x_1^*}{b_1}\right)} * \ldots . * e^{-\left(\frac{x_n - x_n^*}{b_n}\right)} \; if \; |x_n - x_n^*| \leq b_i, i = 1,2, \ldots, n\right) \\ 0 \; otherwise \end{cases} \tag{2.6}$$

2.7.4 Defuzzifier

Defuzzifier mengembalikan hasil perhitungan *fuzzy* (himpunan *fuzzy*) menjadi variabel sesuai rentangnya di dunia nyata. Sama dengan *fuzzifier*, *defuzzifier* juga menggunakan fungsi keanggotaan untuk memetakan nilai himpunan *fuzzy* menjadi *variable* nyata. Terdapat beberapa metode *defuzzifier*, 3 diantaranya yaitu:

a. *Center of gravity defuzzifier*. *Center of gravity* yang dinyatakan dengan y*, menunjukan pusat area yang diliputi oleh fungsi keanggotaan.
b. *Center average dufuzzifier*. *Center average* menunjukan *weight average* dari titik tengah (*center*) masing-masing fungsi keanggotaan.

c. *Maximum defuzzifier*. *Maximum defuzzifier* memilih nilai tertinggi sebagai y*. Ada 3 pilihan, *smallest of maxima*, *largest of maxima* atau *mean of maxima*.

2.8 Kontroller PID

Setiap kekurangan dan kelebihan dari masing – masing kontroler P, I, D dapat saling menutupi dengan menggabungkan ketiganya secara paralel. Oleh karena itu, Kontroler PID merupakan kontroler berumpan balik terdiri dari 3 jenis pengaturan yang saling dikombinasikan, yaitu P *Controller* P (*Proportional*), *Controller* D (*Derivative*), dan *Controller* I (*Integral*) yang masing – masing secara keseluruhan bertujuan untuk mempercepat reaksi sebuah sistem, menghilangkan *offset*, dan menghasilkan perubahan awal yang besar.

Kontroler PID merupakan jumlahan dari *output* kontroler proportional, *output* kontroler integral dan *output* kontroler derivatif. Karakteristik kontroler PID sangat dipengaruhi oleh kontribusi besar ketiga parameter dari P, I dan D. Penyetelan konstanta Kp, Ti, dan Td akan mengakibatkan penonjolan sifat dari masing – masing elemen. Satu atau dua dari ketiga konstanta tersebut dapat disetel lebih menonjol dibanding yang lain. Konstanta yang menonjol itulah akan memberikan kontribusi pengaruh pada respon sistem secara keseluruhan. Parameter-parameter tersebut tidak bersifat independen, sehingga pada saat salah satu nilai konstantanya diubah, maka mungkin sistem tidak akan bereaksi seperti yang diinginkan. [12]

Pengendali proporsional Kp akan memberikan efek mengurangi waktu naik tetapi tidak menghapus kesalahan keadaan tunak. Pengendali integral Ki akan memberikan efek menghapus kesalahan keadaan tunak tetapi berakibat memburuknya tanggapan transient. Pengendali derivatif Kd akan memberikan efek meningkatnya stabilitas sistem, mengurangi lewatan maksimum dan menaikkan tanggapan fungsi trasnfer . Efek dari setiap pengendali dalam sistem *close loop* dapat dilihat pada tabel berikut ini :

Tabel 2.1 Pengaruh K_p, K_i, K_d pada respon sistem

Respon *Close-Loop*	*Rise Time*	*Overshoot*	*Setting Time*	*SS Errror*
K_p	Turun	Naik	Perubahan Kecil	Turun
K_i	Turun	Naik	Naik	Hilang
K_d	Perubahan Kecil	Turun	Turun	Perubahan Kecil

Dari Tabel di atas dapat diketahui bahwa pengendali proporsional akan mengurangi waktu naik, meningkatkan persentase lewatan maksimum dan mengurangi keadaan tunak. Sedangkan pengendali proporsional derivatif mereduksi lewatan maksimum dan waktu turun. Selain itu, pengendali proporsional integral menurun pada waktu naik, meningkatkan lewatan maksimum dan waktu turun dan akan menghilangkan kesalahan keadaan. Salah satu permasalahan terbesar dalam desain kontroler PID yaitu masalah *tuning* untuk menentukan nilaik Ki, Kp, dan Kd yang pas. Metode –metode tuning dilakukan berdasarkan model matemetika *plant* sistem. Jika model tidak diketahui, maka dilakukan eksperimen terhadap sistem. Bisa juga pakai sistem *try* dan *error*. Kontroler PID ini merupakan jenis kontroller yang paling populer digunakan yang banyak diterapkan di dunia industri. Luasnya penggunaaan kontroler PID pada dasarnya dilatarbelakangi oleh beberapa hal diantaranya:

a. Kesederhanaan struktur kontrol. Selain hanya ada 3 parameter utama yang perlu diatur, pengaruh perubahan setiap parameter PID terhadap dinamika pengontrolan secara intuitif mudah dipahami oleh operator.
b. Kontrol PID memiliki sejarah yang panjang yang telah digunakan jauh sebelum era digital berkembang (yaitu sekitar tahun 1930 – an).
c. Kontrol PID dalam banyak kasus telah terbukti menghasilkan unjuk kerja relatif memuaskan, baik digunakan sebagai sistem regulator (sistem kontrol dengan *set point* konstan dan beban cenderung berubah – ubah) maupun sebagai sistem servo

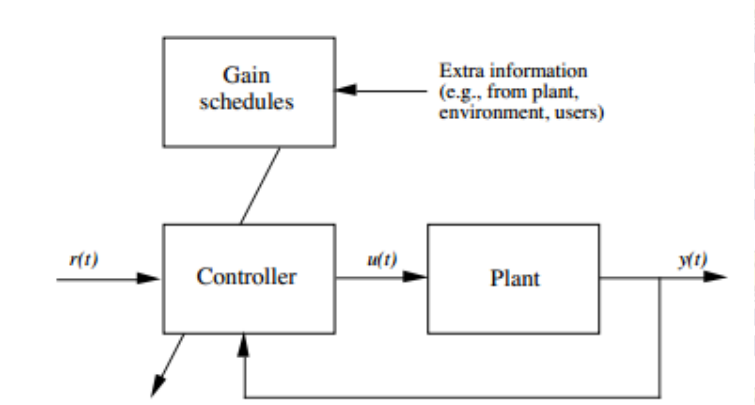
(sistem kontrol dengan *set point* yang berubah dan beban cenderung konstan).

2.9 Kontroler *Fuzzy* PID *Gain Scheduling*

Kontrol PID merupakan algoritma kontrol yang banyak digunakan di industri proses karena bentuknya yang sederhana dan mudah diimplementasikan. Pada kondisi operasi tertentu (seperti misalnya sering terjadi gangguan pada proses atau parameter proses yang berubah-ubah), parameter kontrol ini harus sering di-*tune* agar kinerjanya tetap baik. Salah satu teknik dalam sistem kontrol yang sering dilakukan untuk mengatasi permasalahan ini adalah dengan menggunakan metode *PID Gain Scheduling*, dimana parameter kontrol diubah secara otomatis jika terjadi perubahan kondisi operasi yang menyebabkan kinerja kontrol menurun.

Dalam rangka memberikan contoh penggunaan *fuzzy logic* di bidang kontrol proses, maka selanjutnya akan dibahas salah satu metode PID *gain scheduling* dengan menggunakan *fuzzy logic*. Metode ini dinamakan *Fuzzy PID gain scheduling*. [10]

Pada aplikasi ini, *fuzzy* berfungsi menghitung parameter kontrol PID (Kp, Ti dan Td), berdasarkan kondisi sinyal *error* (E) dan perubahan parameter luar (ΔE).Secara umum, diagram kontrol *fuzzy* PID *gain scheduling* dapat digambarkan seperti berikut.



Gambar 2.10 Diagram Kontrol *Fuzzy Gain Scheduling* [11]

Gambar 2.10 merupakan diagram kontrol sistem pengaturan *Fuzzy Gain Scheduling*. Pada diagram tersebut bisa dilihat bahwa struktur kontrol merupakan struktur konvensional tetapi bedanya kontroler ini dapat diubah parameter kontrolnya dengan menggunakan *Gain Schedules*. *Gain schedules* ini mendapatkan *input* dari informasi ekstra dari luar. Informasi ekstra ini dapat berupa informasi perubahan pada *plant* atau informasi gangguan pada *plant* yang dapat merubah fungsi alih dari sistem.

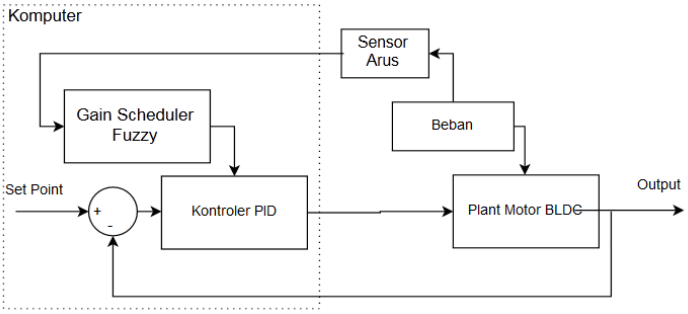
Pada tugas akhir ini kontroler yang digunakan merupakan kontroler PID yang sudah biasa dipakai pada dunia-dunia industri. Kontroler PID ini nantinya bisa diubah parameter kontrolnya yang berupa nilai K_p, K_i, dan K_d. Parameter ini diubah dengan *gain schedules* yang pada tugas akhir ini berupa *fuzzy*. *Fuzzy* ini nantinya akan mendapatkan informasi ekstra berupa informasi dari rem. Informasi rem ini berupa informasi arus yang masuk pada belitan rem magnetik.

Halaman ini sengaja dikosongkan

BAB 3
PERANCANGAN SISTEM

3.1 Gambaran Umum Sistem

Pada tugas akhir ini digunakan sebuah sistem yang telah dirancang dan dibangun sendiri. *Plant* utama yang digunakan pada tugas akhir ini merupakan motor BLDC. *Plant* ini direncanakan dan dibuat oleh team BLDC. Team BLDC ini dibentuk karena kami memiliki tujuan untuk membangun sebuah *plant* baru yang digunakan sebagai bahan untuk tugas akhir dan mungkin nantinya bisa dimanfaatkan sebagai penelitian lebih lanjut. Oleh karena itu plant motor BLDC yang telah direalisasikan ini akan kami beri nama BLDC V-1. Untuk membuat sistem yang *close loop* maka dibutuhkan sebuah sensor, maka dari itu dalam sistem ini terdapat juga sebuah sensor *rotary encoder* yang berfungsi menghitung nilai rpm. Setelah itu agar model dari sistem lebih bervariasi maka diberikanlah sebuah sistem pembebanan yang menggunakan rem magnetik. Oleh karena itu, terbentuklah sebuah sistem *close loop* yang nantinya akan diberikan sebuah kontroler yang akan mengatur keluaran kecepatan dari motor BLDC V-1. Berikut blok diagram dari sistem pengaturan kecepatan motor BLDC V-1.



Gambar 3.1 Blok Diagram Sistem Pengaturan Kecepatan Motor *Brushless DC* (BLDC).

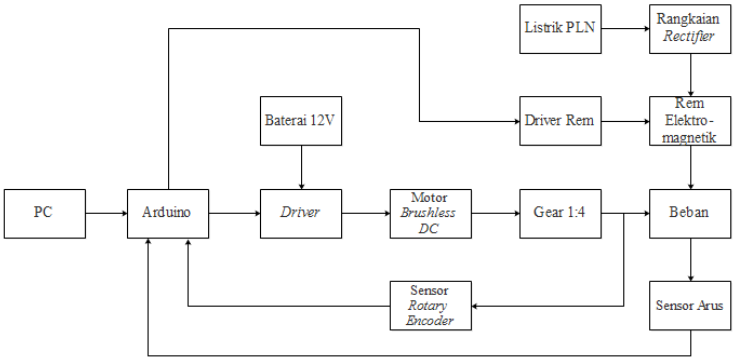
Selain perangkat keras berupa motor BLDC dan rem elektromagnetik, ditambahkan pula beberapa komponen pendukung, seperti *driver* untuk motor BLDC dan *driver* untuk rem elektromagnetik.

Selain itu, dilengkapi pula sensor arus untuk mengetahui besaran arus yang diberikan pada komponen rem elektromagnetik sebagai efek pembebanan pada motor BLDC. Mikrokontroler Arduino juga akan dipakai pada sistem ini sebagai perantara antara sensor-sensor dan *driver* dengan komputer.

Dalam sistem ini, motor BLDC V-1 merupakan komponen yang dikontrol untuk mencapai *output* yang diinginkan. Demi tujuan tersebut, digunakan kontroler berbasis *Fuzzy PID Gain Scheduling* guna mencapai performansi yang diinginkan. Kontroler ini akan mengeluarkan sinyal kontrol yang kemudian dikirimkan ke arduino yang datanya aka diteruskan ke ESC untuk menggerakkan motor BLDC. Dengan begitu, *output* yang dihasilkan diharapkan bisa sesuai dengan referensi yang diinginkan

3.2 Perancangan Perangkat Keras

Pada tahap perancangan keras, terdapat 2 jenis perancangan yang dilakukan, yaitu perancangan mekanik dan elektronik. Perancangan mekanik merupakan perancangan komponen utama pada *plant*, yaitu berupa motor BLDC V-1 dan rem elektromagnetik. Sedangkan perancangan elektronik merupakan perancangan untuk kontroler, *driver* dan rangkaian sensor yang akan digunakan pada *plant* ini. Kontroler akan diprogram didalam PC dan sebagai perantara komputer dengan *plant*, digunakan mikrokontroler arduino yang juga berfungsi sebagai perangkat akuisisi data.



Gambar 3.2 Konfigurasi Perangkat Keras Sistem Pengaturan Kecepatan Motor *Brushless DC* (BLDC).

Arduino akan menerima data dari sensor dan mengirimkannya kepada komputer. Selain itu, digunakan digunakan pula rangkaian *driver* untuk menggerakan motor BLDC V-1 dan memberi *input* arus pada rem elektromagnetik. Terdapat pula rangkaian sensor yang berfungsi mengukur *input* arus yang masuk ke dalam rem elektromagnetik dan sensor kecepatan motor BLDC V-1 berupa sensor *rotary encoder*. Konfigurasi perangkat keras pada *plant* dapat dilihat pada Gambar 3.2.

3.2.1 Perancangan Mekanik

Pada *plant* sistem pengaturan kecepatan motor BLDC V-1, terdapat beberapa komponen utama yang digunakan, antara lain motor BLDC V-1 sebagai tenaga penggerak dan objek yang dikontrol. Selain itu, terdapat rem elektromagnetik yang digunakan untuk memberikan efek pembebanan pada motor BLDC V-1. Rem elektromagnetik diberikan *input* arus DC yang berbentuk PWM. Untuk lebih jelasnya, penjelasan mengenai konstruksi motor BLDC V-1 dan rem elektromagnetik dapat dilihat pada subbab di bawah ini.

3.2.1.1 Motor Brushless DC

Motor Brushless DC (BLDC) dipilih karena efisiensi dan konstruksinya yang tidak menggunaakan *brush* atau sikat. Motor BLDC yang digunakan merupakan motor BLDC yang digunakan pada pesawat *aeromodelling*. Motor BLDC jenis ini merupakan miniatur dari motor BLDC penggerak kendaraan listrik dikarenakan konstruksinya yang lebih kecil. Motor yang digunakan merupakan berasal dari produsen RCTimer dengan nomor seri HP2212-1000KV. Bentuk fisik motor BLDC yang digunakan pada *plant* ini dapat dilihat pada Gambar 3.3. Sedangkan spesifikasi motor BLDC dapat dilihat pada Tabel berikut di bawah ini.

Tabel 3.1 Spesifikasi Motor BLDC RCTimer HP2212-1000KV. [3]

Parameter	Nilai	
Kekuatan Magnet	40 µH	
Berat Motor	59,06 gram	
KV	1002,7 rpm/volt	
Kecepatan Motor	Tanpa Beban	11.130 rpm
	Beban Maksimal	6.840 rpm

Tabel 3.1 Lanjutan

Parameter	Nilai	
Input Arus	Tanpa Beban	0,7 Ampere
	Beban Maksimal	13,9 Ampere
Power Motor	Beban Minimal	44,4 Watt
	Beban Maksimal	154,29 Watt
Input Baterai	Lithium Polimer 2S-4S	

Gambar 3.3 Motor *Brushless* DC Tipe *Outrunner* dengan Tipe RCTimer HP2212-1000KV.

3.2.1.2 Rem Elektromagnetik

Rem elektromagnetik pada *plant* ini berguna sebagai pembebanan pada motor. Medan elektromagnetik dari rem ini dihasilkan oleh beberapa kumparan yang dihubungkan secara seri dan paralel dan diberikan masukan arus DC. Arus DC yang masuk ke dalam rem elektromagnetik berbentuk PWM yang *duty cycle*-nya diatur oleh *driver* dan Arduino. Adapun sumber tegangannya didapat dari jala-jala PLN yang diberikan trafo *step-down* dari 220V menjadi 36V. lalu *output* pada trafo akan disearahkan melalui rangkaian *rectifier*.

Rem elektromagnetik ini terbuat dari 8 kumparan, terdiri dari 4 kumparan di setiap sisinya, yang disusun secara seri dan paralel. Maksudnya 4 kumparan pada satu sisi akan dirangkai secara seri begitu juga sisi lainnya. Lalu kedua sisi tersebut dihubungkan secara paralel. Diantara celah kedua sisi kumparan dipasang piringan aluminium. Piringan aluminium tersebut lalu dipasang ke dalam sebuah *shaft* yang selanjutnya dikopel dengan motor BLDC V-1. Perbandingan *gear* yang

digunakan untuk mengkopel motor BLDC V-1 dengan *shaft* tersebut adalah 1:4.

Kumparan ini terbuat dari 8 buah baut dan 16 buah mur. Lalu pada baut dilapisi kertas sebagai isolator. Selanjutnya kawat tembaga dililit pada kertas tersebut hingga jumlah lilitan yang telah ditentukan pada Persamaaan 3.1. Jumlah lilitan tiap kumparan dihitung berdasarkan rumus berikut ini:

$$N = \frac{44}{d} \times V = \frac{44}{1,4} \times 36 = 1131 \qquad (3.1)$$

N : Jumlah lilitan tiap sisi

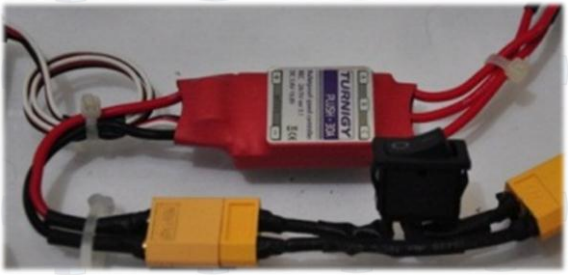
d : Diameter kumparan (cm)

V : Tegangan *input* (volt)

Berdasarkan hasil perhitungan lilitan yang diperukan ada 1131 lilitan dikarenakan tegangan *input* yang digunakan adalah 36 Volt. Jumlah tersebut masih harus dibagi 4 dikarenakan tiap sisi rem yang mempunyai 4 kumparan. Jadi, tiap kumparan memiliki 283 lilitan.

3.2.1.3 Driver Motor BLDC V-1

Driver pada motor BLDC kali ini menggunakan *Electronic Speed Controller* (ESC) yang merupakan driver yang biasa digunakan pada RC motor atau pesawat. ESC ini memiliki *input* berupa PWM yang memiliki frekuensi sekitar 50 Hz. PWM yang dapat diterima oleh ESC tidak pada seluruh *duty cycle*. Kebanyakan ESC biasanya hanya dapat menerima *input* pwm yang memiliki *duty cycle* 5% hingga 10%. Berikut betuk fisik dari ESC yang digunakan.



Gambar 3.4 ESC.

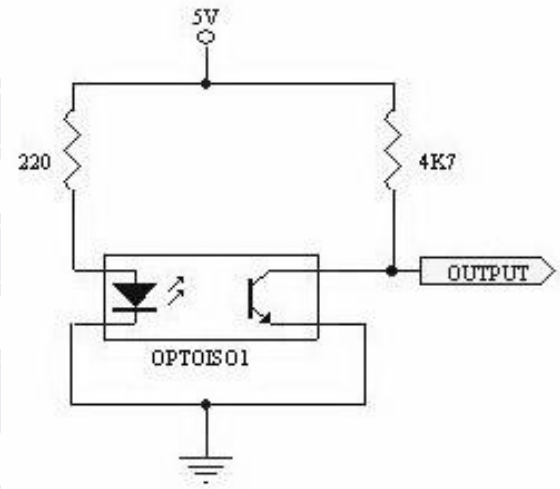
3.2.2 Perancangan Elektronik

Perancangan elektronik ini meliputi desain layout rangkaian PCB serta pengkabelan. Rangkaian elektronik pada *plant* ini meliputi rangkaian *driver* motor BLDC, *driver* rem elektromagnetik serta rangkaian sensor kecepatan *rotary encoder* dan sensor arus. Selain itu, digambarkan pula rancangan pengkabelan pada mikrokontroler Arduino.

3.2.2.1 Rangkaian Sensor *Rotary Encoder*

Untuk mengukur kecepatan sebuah motor dibutuhkan sebuah sensor. Ada berbagai jenis sensor yang dapat digunakan dalam mengukur kecepatan sudut pada motor. Contohnya saja ada *tachogenerator* yang cara kerjanya adalah dengan cara mengkopel poros sebuah motor dengan sebuah motor DC yang difungsikan sebagai generator, lalu *output* generator tersebutlah yang akan berubah-ubah tegangannya seiring berubahnya kecepatan pada motor DC. Lalu ada lagi *hall* sensor yang menggunakan *hall effect* sebagai prinsip kerjanya, *hall effect* merupakan sebuah prinsip yang initinya apa bila sebuah magnet yang didekatkan dengan sebuah plat logam maka pada logam tersebut akan timbul arus yang akan mengalir. Dan pada tugas akhir kali ini digunakan sensor *rotary encoder* yang mengunakan *phototransistor*. *Phototransistor* ini akan menerima cahaya *infrared* yang akan terus terpancar dari sebuah *emitter*. Kedua alat ini yaitu *emitter* dan *phototransistor* ini ada di dalam sebuah alat yaitu *slotted optocoupler*. Lalu diantara sela-sela *optocoupler* akan ditutupi dengan sebuah piringan yang memiliki lubang yang relatif kecil yaitu 1 mm. piringan ini berdiameter 5 cm akan berputar sehingga ketika lubang dari piringan melewati *optocoupler* maka sebuah *pulse high* akan terpancar dari *output* rangkaian.

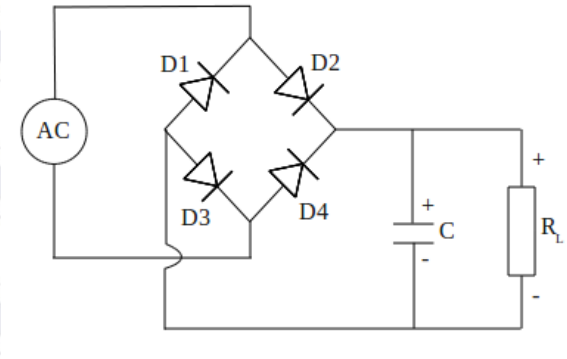
Rangkaian untuk *phototransistor* memiliki struktur yang persis sama dengan *transistor* biasa begitu juga dengan cara kerjanya. Perbedaanya apabila *transistor* biasa memakai arus sedangkan *phototransistor* memakai cahaya sebagai *base*. Berikut rangkaian dari sensor *rotary encoder* yang digunakan pada alat ini.



Gambar 3.5 Rangkaian Sensor *Rotary Encoder*.

3.2.2.2 Rangkaian Rectifier

Pada sistem ini dibutuhkan arus DC yang cukup besar yaitu 36 Volt, dan untuk merealisasikan hal itu maka digunakanlah energi listrik yang berasal dari PLN yang tegangannya akan diturunkan menjadi 36 Volt menggunakan *transformator* lalu disearahkan dengan rangkaian *rectifier* yang ada di bawah ini.



Gambar 3.6 Rangkaian *Rectifier*.

Dengan parameter pada Gambar 3.6 memiliki nilai AC = 36 Volt, nilai C = 300 µF/100 V, nilai tiap dioda D1, D2, D3, dan D4 sama dengan 0,7 Volt, dan yang terakhir merupakan beban dari gulungan sebesar 5,5 Ohm.

Pada rangkaian *rectifier* ini terdiri dari 4 buah dioda dan satu buah kapasitor. 4 buah dioda berfungsi sebagai merubah fasa negatif pada sumber listrik menjadi fasa positif sehingga pada satu periode yang seharusnya berbentuk sinusoidal berubah menjadi sinusoidal yang memiliki tegangan positif di bagian negatifnya. Setelah di searahkan dengan 4 buah dioda didapatkan sebuah sinyal yang masih terdapat *ripple. Ripple* ini diredam dengan menggunakan kapasitor agar sinyalnya dapat menyerupai arus DC.

3.2.2.3 Rangkaian Driver Rem Elektromagnetik

Rem eektromagnetik yang digunakan pada *plant* ini dapat dikendalikan kekuatan medan magnetnya. Hal ini dapat dilakukan dengan mengubah besarnya arus yang akan masuk ke kumparan. Cara termudah untuk merubah arus pada kumparan adalah dengan mengatur tegangan masukan pada kumparan karena dengan berubahnya nilai tegangan masukan pada kumparan maka tentunya nilai arus masukan pada kumparan akan ikut berubah menyesuaikan dengan tegangan. Hal ini berjalan dengan sesuai dengan hukum Ohm yaitu V = I.R, karena R pada kumparan akan selalu bernilai sama maka nilai tegangan akan selalu sebanding dengan nilai arus. Hal inilah yang kami manfaatkan untuk dapat merubah-ubah besarnya medan magnet pada kumparan.

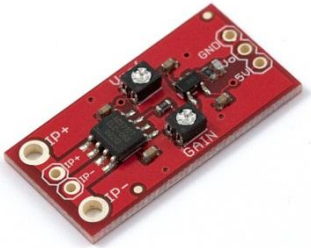
Ada beberapa cara untuk dapat mengatur tegangan. Misalnya saja rangkaian pembagi tegangan dan dengan membuat tegangan menjadi sebuah sinyal *Pulse Width Modulation* (PWM). Pada tugas akhir kali ini untuk mengatur tegangan masukan pada kumparan digunakan metode yang menggunakan sinyal PWM. Yaitu dengan dengan membuat sebuah rangkaian yang dapat merubah tegangan DC menjadi sinyal PWM yang memiliki amplitudo sebesar tegangan yang diberikan.

3.2.2.4 Sensor Arus

Untuk mendapatkan informasi seberapa besar beban yang dibebankan pada piringan alumunium digunakan sebuah sensor arus yang dipasangkan seri pada lilitan rem elektromagnetik.

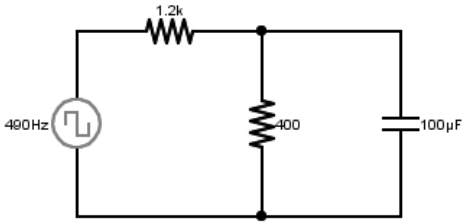
Sensor arus yang akan digunakan pada sistem pengaturan motor BLDC ini adalah sensor ACS712 yang merupakan sensor arus yang merubah *input* berupa arus menjadi tegangan. Spesifikasi dari sensor

arus ACS712 yang digunakan kali ini memiliki *input* berupa arus yang memiliki *range* sebesar -5A sampai dengan 5A dan *output* yang keluar dari sensor arus ini merupakan tegangan dari 0 sampai 5V.



Gambar 3.7 Sensor Arus ACS712.

Hasil dari sensor ini merupakan tegangan 0V sampai dengan 5V. Karena arus yang masuk ke *input* pada sensor merupakan arus yang berbentuk PWM, maka *output* pada sensor arus juga merupakan tegangan yang berbentuk PWM. Oleh karena itu Arduino tidak dapat membaca nilai rata-rata tegangan *output* pada sensor. Agar Arduino dapat membaca tegangan rata-rata maka diberikan kapasitor pada *output* sensor agar dapat menghasilkan rata-rata tegangan. Rangkaian ini dapat dilihat pada Gambar 3.8.



Gambar 3.8 Rangkaian RC *Filter*.

3.2.2.5 Pengkabelan Mikrokontroler Arduino

Arduino memiliki 14 *input output* digital dan 6 *input output* analog. Pada Arduino Uno juga terdapat pin yang menggunakan sinyal PWM sebagai *input* dan *output* yaitu pin 3,5,6,9,10,11. Pada tugas akhir ini digunakan beberapa buah pin *input output* dengan rincian sebagai berikut:

a. Pin A0 : *input* analog dari potensio
b. Pin 7 : *input encoder*
c. Pin 9 : *output* PWM untuk motor BLDC
d. Pin 11 : *output* PWM untuk rem magnetik

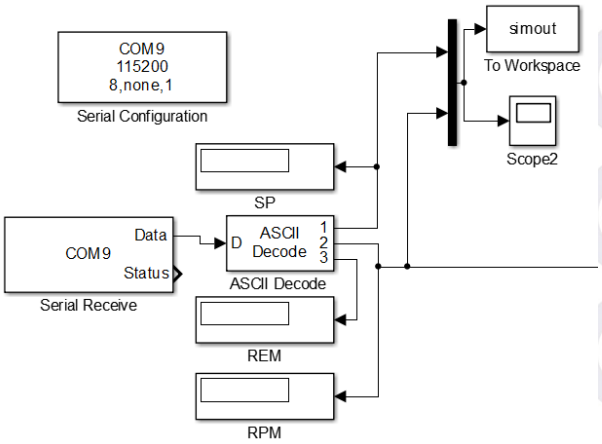
Selain pin-pin di atas tentunya dibutuhkan juga pin 5 Volt dan ground karena setiap sensor dan motor mebutuhkan *ground*.

3.3 Perancangan Perangkat Lunak

Dalam sistem pengaturan motor BLDC, beberapa perangkat lunak harus digunakan untuk proses pengambilan data, perancangan kontroler, dan pengiriman data. Perangkat lunak yang digunakan yaitu MATLAB dan *software* Arduino.

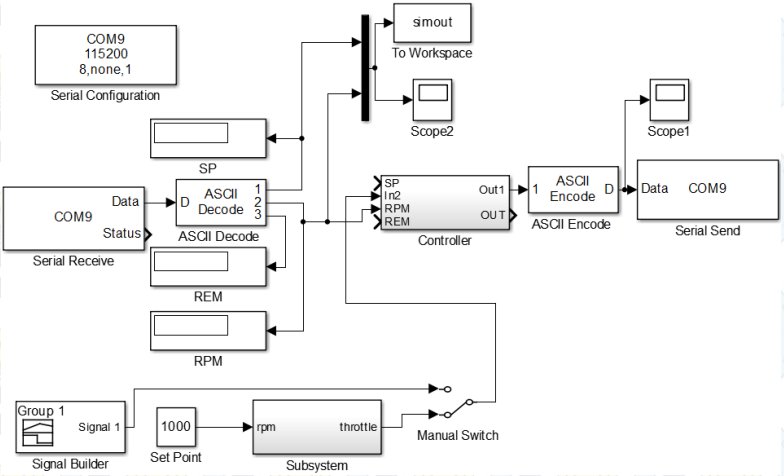
3.3.1 Perangkat Lunak Matlab

Perangkat lunak digunakan sebagai antarmuka antara mesin dan pengguna, biasa disebut *Human Machine Interface* (HMI). HMI yang digunakan pada tugas akhir ini adalah MATLAB2013a. Perangkat lunak ini digunakan pada proses pengambilan data dan pengiriman data. Dengan menggunakan komunikasi serial dengan Arduino, Simulink MATLAB dapat mengolah data dari blok *serial receive* dan mengirimkannya kembali melalui blok *serial send*. Contoh diagram blok Simulink MATLAB untuk pengambilan data *open loop* ditunjukkan pada gambar berikut.



Gambar 3.9 Blok Simulink *Open Loop*.

Setelah itu diagram blok Simulink yang dipakai pada saat identifikasi bisa dilihat pada gambar berikut.



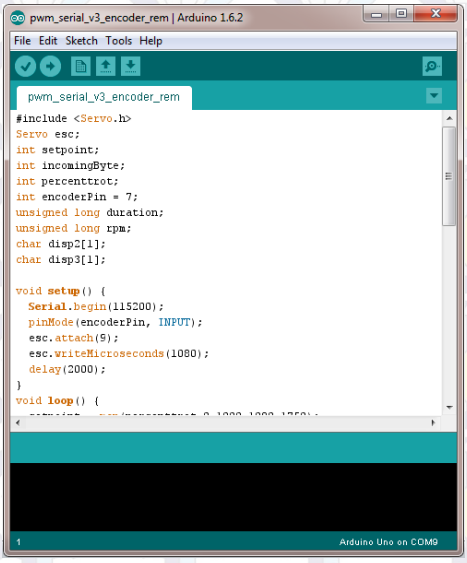
Gambar 3.10 Blok Simulink Identifikasi.

3.3.2 Perangkat Lunak Arduino

Pada sistem pengaturan motor BLDC V-1 ini, Arduino digunakan untuk *interface* antara sensor, potensio, *output* ke motor BLDC dengan Simulink MATLAB. Artinya, Arduino ini berfungsi sebagai media pengirim data yang menghubungkan antara sensor dan *plant* dengan Simulink MATLAB. Data yang diterima oleh *serial receive* pada Simulink MATLAB merupakan data ASCII. Dari data ASCII yang dikirim ini terdiri dari 3 variabel yaitu nilai kecepatan pada sensor yang memiliki satuan rpm, data pada rem magnetik, lalu data *set point* yang diinginkan yang dikirim dalam bentuk *throttle*. *Throttle* ini memiliki *range* yang berkisar antara 0 sampai 1000 yang menunjukkan nilai PWM dari *duty cycle* 0 sampai 100%. Data dari rem memiliki range dari 1-999 yang lalu dirubah menjadi data dari 1 sampai 254. Data dari 1 sampai 254 ini akan dimasukkan pada pin 11 sebagai *output* PWM dari *duty cycle* sekitar 1% sampai 99%. Nilai ini dibatasi pada 1 dikarenakan jika PWM diberi nilai 0 ditakutkan terjadi kerusakan pada komponen.

Selanjutnya data dari kecepatan motor BLDC V-1 merupakan data hasil perhitungan yang menggunakan fungsi PulseIn( ). Fungsi ini akan mengembalikan sebuah nilai pada sebuah variabel yang telah di

deskripskan. Pertama-tama fungsi ini akan membaca nilai *high* atau *low* pada sebuah pin, pada sistem ini dipakai pin 7 dan yang dibaca pada pin merupakan *pulse high*. Lalu apabila pin 7 membaca nilai *high*, *timer* akan berjalan dan akan menunggu hingga pin 7 membaca nilai *low*. Apabila pin 7 membaca nilai *low* maka *timer* akan berhenti dan variabel yang telah disebutkan akan mendapatkan nilai waktu lamanya pulsa berada pada keadaan *high*. Disinilah data kecepatan motor bisa didapatkan dengan mendapatkan seberapa lama waktu yang dibutuhkan piringan dari lubang satu ke lubang yang lain. Dari waktu tersebut dapat didapatkan nilai kecepatan motor dalam satuan rpm. Dan yang terakhir data dari pembacaan arus yang berasal dari sensor arus pada rem magnetik. Data dari sensor arus ini merupakan data berupa tegangan dari *range* 0 V sampai 1,25 V. data ini akan masuk ke Ardunio lalu dibaca dengan menggunakan pin analog. Data yang dihasilkan akan berupa data berupa nilai sebesar 0 sampai 245. Setelah itu data nilai sebesar 0 sampai 245 akan dikalikan 4. Untuk membeuat sebuah *script* yang telah dirancang digunakan Arduino IDE. Program yang telah dibuat bisa di *upload* pada Arduino denggan menggunakan Arduino IDE. Berikut tampilan dari Arduino IDE.



Gambar 3.11 Tampilan IDE Arduino.

3.4 Identifikasi dan Pemodelan Sistem

Pada pemodelan sistem di tugas akhir kali ini digunakan identifikasi dinamis. Metode dinamis ini dilakukan dengan cara memberikan *input* berapa sinyal *Pseudorandom Binary Sequence* (PRBS). Setelah itu *output* kecepatan dari akan diidentifikasi dengan menggunakan metode ARX yang ada pada *toolbox* pada MATLAB. Setelah hal ini dilakukan didapatkanlah model matematika dari *plant.*

3.4.1 Metode dan Pembebanan Sistem

Respons *plant* dapat dilihat dari beberapa cara. Salah satu cara yaitu dengan memberikan pembebanan. Pada *plant* motor BLDC V-1 pembebanan dilakukan dengan menggunakan rem elektromagnetik. Pembeban dilakukan untuk mendapatkan model sistem dengan beban yang ditentukan. Pada sistem ini rem magnetik di berikan rem magnetik dari mulai dari *input* 0, 20%, 40%, 60%, dan 80% *duty cycle*. Beban-beban tersebut disebut dengan beban ringan, sedang, berat, sangat berat. Berikut hubungan *input output* persentasi *duty cycle* dengan *output* tegangan pada rem magnetik.

Tabel 3.2 Tabel Hubungan *Duty Cycle* dengan Tegangan.

Beban	*Duty Cycle* PWM	Tegangan (Volt)	Tegangan *Output* Sensor Arus (Volt)	Nilai ADC Sensor Arus
Tanpa Beban	0%	0,000	0,084	12
Ringan	20%	2,054	0,905	190
Sedang	40%	4,010	1,870	416
Berat	60%	7,330	2,870	624
Sangat Berat	80%	11,94	3,939	852

Dari tabel ini dapat dilihat bahwa semakin besar nilai *duty cycle* dari PWM maka akan semakin besar pula *output* tegangan yang

dihasilkan. Hal ini dikarenakan sinyal pwm akan dikalikan dengan amplitude dari tegangan DC 36 V. Dan hasil dari perkalian akan didapatkan sebuah tegangn yang berbentuk seperti PWM tetapi memiliki amplitude sebesar 36 V. Dan pada saat beban ringan nilai tegangan bernilai 2,054 V karena tegangan ini merupakan tegangan rata-rata yang dikeluarkan oleh teganan PWM yang memiliki amplitude 36 V

3.4.2 Metode Identifikasi dan Pemodelan

Identifikasi pada motor BLDC V-1 dilakukan dengan menggunakan identifikasi dinamis. Metode yang dipakai untuk identifikasi merupakan metode ARX. Hal ini dilakukan dengan cara memberikan *input* dengan menggunakan sinyal PRBS. Pada tugas akhir ini sinyal PRBS yang digunakan merupakan sinyal kotak yang memiliki batas bawah dan batas atas. Karena batasan kerja dari tugas akhir yang diteliti telah ditentukan sejak awal yaitu dari kecepatan 750 sampai 1750 maka sinyal *input* yang diberikan pada motor ini bernilai 74 sampai 147.

3.4.2.1 Tidak Diberi Beban

Pada tahap ini motor sama sekali tidak diberi beban sehingga ini merupkan pemodelan motor yang hanya memiliki beban berupa piringan alumunium yang menempel pada *shaft*.

Setelah didapatkan data dari *input* dan *output* motor maka tahap selanjutnya adalah melakukan identifikasi dari data yang telah didapatkan. Pada identifikasi dengan metode ARX kali ini hanya dilakukan dari orde 1 orde 2, dan orde 3. Metode ini dilakukan dengan *toolbox identification system* yang ada pada program MATLAB. Berikut hasil fungsi alih yang didapatkan setelah dilakukan identifikasi menggunakan metode ARX.

Tabel 3.3 Fungsi Alih Saat Tidak diberi Beban.

Orde	Fungsi Alih *Plant*	RMSE
1	$G(z) = \frac{0.02026}{z - 0.9983}$	0,0827
2	$G(z) = \frac{0.003406z + 0.02739}{z^2 - 1.006z + 0.008015}$	0,0797

Tabel 3.3 Lanjutan.

Orde	Fungsi Alih *Plant*	RMSE
3	$G(z) = \frac{-0.003329 + 0.01711z + 0.01045}{z^3 - 1.006z^2 + 0.0139z - 0.005846}$	0,0777

Setelah didapatkan semua fungsi alih pada orde 1, orde 2, dan orde 3. Yang dilakukan selanjutnya adalah menghitung *error*. Perhitungan *error* dilakukan dengan menggunakan metode *Root Mean Square Error* (RMSE). Setelah dilakukan perhitungan RMSE didpatkanlah bahwa pada fungsi alih untuk motor pada kondisi tanpa beban orde yang memiliki *error* terkecil merupakan fungsi alih orde 3.

3.4.2.2 Fungsi Alih Saat Diberi Beban

Pada Tahap ini yang dilakukan adalah sama seperti pada tahap sebelumnya yaitu pengambilan data dinamis lalu identifikasi, lalu penghitungan *error*. Setelah hal itu dilakukan Semua maka didapatkanlah fungsi alih sebagai berikut.

Tabel 3.4 Fungsi Alih Saat Diberi Beban Ringan.

Orde	Fungsi Alih *Plant*	RMSE
1	$G(z) = \frac{0.02197}{z - 0.9981}$	0,2072
2	$G(z) = \frac{0.012924z + 0.01613}{z^2 - 0.99725z - 0.0002356}$	0,2104
3	$G(z) = \frac{0.006063z^2 + 0.001305z + 0.01488}{z^3 - 0.9978z^2 + 0.001073z - 0.00132}$	0,2008

Pada identifikasi motor BLDC dengan beban ringan ini dapat dilihat bahwa fungsi alih yang memiliki *error* terkecil ada pada fungsi alih orde 3. Dapat dilihat bahwa error yang terjadi cukup besar. Dan

ternyata setelah diteliti lebih lanjut terdapat data *input* yang hilang karena terjadi *lag* pada komputer.

Tabel 3.5 Fungsi Alih Saat Diberi Beban Sedang.

Orde	Fungsi Alih *Plant*	RMSE
1	$G(z) = \frac{0.02194}{z - 0.9979}$	0,1008
2	$G(z) = \frac{0.06103z + 0.03728}{z^2 - 0.999z - 0.00129}$	0,0951
3	$G(z) = \frac{0.005813z^2 - 0.03686z + 0.0006807}{z^3 - 1.017z^2 + 0.008681z - 0.01009}$	2,7684

Pada identifikasi motor BLDC V-1 dengan beban sedang ini dapat dilihat bahwa fungsi alih yang memiliki *error* terkecil ada pada fungsi alih orde 2. Pada fungsi oleh orde 3 terdapat *error* yang sangat besar, hal ini terjadi karena kesalahan identifikasi sehingga pada respon output hasil pemodelan memiliki respon *output* yang negatif.

Tabel 3.6 Fungsi Alih Saat Diberi Beban Berat.

Orde	Fungsi Alih *Plant*	RMSE
1	$G(z) = \frac{0.02568}{z - 0.9973}$	0,01009
2	$G(z) = \frac{-0.00061976z + 0.03103}{z^2 - 0.9977z - 0.000887}$	0,10030
3	$G(z) = \frac{0.005813z^2 + 0.03686z + 0.0006807}{z^3 - 1.0068 + 0.018681z - 0.01009}$	0,77040

Pada identifikasi motor dengan beban berat ini dapat dilihat bahwa fungsi alih yang memiliki *error* terkecil ada pada fungsi alih orde 1.

Tabel 3.7 Fungsi Alih Saat Diberi Beban Sangat Berat.

Orde	Fungsi Alih *Plant*	RMSE
1	$G(z) = \frac{0.03051}{z - 0.9962}$	0,1408
2	$G(z) = \frac{-0.2031z + 0.2362}{z^2 - 1.097z - 0.1011}$	0,1422
3	$G(z) = \frac{-0.2221z^2 + 0.2442z + 0.004852}{z^3 - 1.099z^2 + 0.1162z - 0.01371}$	0,1235

Pada identifikasi motor dengan beban sangat berat ini dapat dilihat bahwa fungsi alih yang memiliki *error* terkecil ada pada fungsi alih orde 3.

3.4.3 Validasi Model

Dari semua model yang telah didapat didapatkan model yang paling tepat untuk menggantikan motor BLDC V-1 saat simulasi. Model-model ini nantinya akan digunakan sebagai pengujian respon kontroler saat simulasi karena fungsi alih ini memiliki respon yang paling mirip dengan respon motor BLDC V-1. Berikut hasi dari pemodelan fungsi alih di setiap pembebanan.

Tabel 3.8 Fungsi Alih Di Setiap Pembebanan.

Beban	Fungsi Alih *Plant*	RMSE
Tanpa Beban	$G(z) = \frac{-0.003329 + 0.01711z + 0.01045}{z^3 - 1.006z^2 + 0.0139z - 0.005846}$	0,7770
Ringan	$G(z) = \frac{0.006063z^2 + 0.001305z + 0.01488}{z^3 - 0.9978z^2 + 0.001073z - 0.00132}$	0,2008

Tabel 3.8 Lanjutan.

Beban	Fungsi Alih *Plant*	RMSE
Sedang	$G(z) = \frac{0.06103z + 0.03728}{z^2 - 0.999z - 0.00129}$	0,09510
Berat	$G(z) = \frac{0.02568}{z - 0.9973}$	0,01009
Sangat Berat	$G(z) = \frac{-0.2221z^2 + 0.2442z + 0.004852}{z^3 - 1.099z^2 + 0.1162z - 0.01371}$	0,12350

3.5 Perancangan Kontroler *Fuzzy* PID *Gain Scheduling*

Setelah fungsi alih dari sistem telah didapatkan, langkah selanjutnya adalah merancang kontroler untuk masing-masing pembebanan. Kontroler digunakan untuk mengembalikan respon ke nilai *set point* yang diinginkan sekalipun motor BLDC diberi beban. Tahapan desain kontroler ini meliputi perancangan kontroler PID dan perancangan kontroler *fuzzy*.

3.5.1 Perancangan Kontroler PID

Sebelum merancang kontroler *fuzzy* harus dicari dulu kontroler PID yang sesuai untuk setiap pembebanan. Dengan merancang kontroler PID ini maka akan didapatkan nilai fungsi keanggotaan untuk *output* pada kontroler *fuzzy*.

Pada tahap ini direncanakan menggunakan metode Ziegler-Nichols untuk mencari nilai K_p, K_i, dan K_d. Namun setelah PID hasil metode Ziegler-Nichols disimulasikan ternyata tidak menghasilkan *output* yang sesuai dengan keinginan penulis. Oleh karena itu PID yang telah dirancang di sini merupakan PID hasil percobaan penulis sendiri yang telah diatur sedemikian rupa agar memiliki *output* yang sesuai dengan keinginan. Nilai konstanta ini sebenarnya bukan berdasarkan pencarian nilai secara coba-coba melainkan mmiliki dasar juga. Dasar dari pemberian nilai ini memiliki memiliki konsep dasar pengetahuan secara heuristik misalnya apabila respon memiliki *error steady state* lalu apabila pada kontroler deberi konstanta K_i maka *error steady state* akan terhapus, lalu misalnya apa bila terdapat *overshoot* maka nilai K_d harus diberi nilai yang lebih besar sehingga menambah transien pada respon.

Oleh karena itu, setelah semua nilai K_p, K_i, K_d dicari didapatkanlah nilai-nilai berikut.

Tabel 3.9 Nilai K_p, K_i, K_d di Setiap Pembebanan.

Pembebanan	K_p	K_i	K_d
Tanpa Beban	0,165	0,049	0,000
Ringan	0,160	0,050	0,001
Sedang	0,155	0,055	0,005
Berat	0,140	0,060	0,010
Sangat Berat	0,120	0,070	0,020

3.5.2 Perancangan Kontroler *Fuzzy*

Pada tahap perancangan kontroler *fuzzy* terdapat berberapa tahap lagi yaitu penentuan fungsi keanggotaan, penentuan bentuk fungsi keanggotaan, Menentukan fungsi keanggotaan pada *output*.

Kontroler *fuzzy* pada sistem pengaturan motor BLDC V-1 ini berfungsi sebagai pengatur nilai K_p, K_i, K_d pada kontroler PID, sehingga pada sistem ini nilai K_p, K_i, K_d akan menjadi fungsi keanggotaan pada *output* pada *fuzzy*. Pada *output fuzzy* akan dibutuhkan 3 *output* yaitu *output* yang memberikan nilai K_p, K_i, dan K_d. Oleh karena itu, pada fungsi keanggotaan *input* terdapat 3 fungsi keanggotaan yang masing-masing mewakili K_p, K_i, dan K_d. Oleh karena itu, pada *fuzzy* ini terdapat 3 kontroler *fuzzy* yang masing-masing memiliki satu fungsi keanggotaan *input* dan satu fungsi keanggotaan *output*.

3.5.2.1 Rule Base

Karena tujuan akhir dari kontroler *fuzzy* hanya mengeluarkan nilai K_p, K_i, dan K_d yang sesuai saat pembebanan kondisi tertentu maka *rule base* pada ada 5 pernyataan di setiap *gain* K_p, K_i, dan K_d. Misalkan saja pembebanan direpresentasikan dengan variabel X dan *output*

direpresentasikan dengan variabel Y dan anggap saja nilai K_p pada saat tanpa beban bernialai 0,165 dan saat beban ringan bernilai 0,16 dan seterusnya. Maka *rule base* yang dibuat pada kontroler *fuzzy* akan terlihat seperti berikut.

If X = 12 Then Y = 0,165
If X = 190 Then Y = 0,160
If X = 416 Then Y = 0,155
If X = 624 Then Y = 0,140
If X = 852 Then Y = 0,120

Ini merupakan *rule base* pada pencarian nilai K_p, Bisa dilihat bahwa pada nilai K_p semakin besar pembebanan maka semakin kecil nilai K_p yang diberikan. Setelah *rule base* pada K_p telah ditentukan maka yang selanjutnya adalah membuat *rule base* pada nilai K_i. Berikut ini merupaka *rule base* pada pencarian nilai K_i.

If X = 12 Then Y = 0,049
If X = 190 Then Y = 0,050
If X = 416 Then Y = 0,055
If X = 624 Then Y = 0,060
If X = 852 Then Y = 0,070

Dari *rule base* untuk pencarian nilai K_i di setiap pembebanan ini dapat dilihat bahwa semakin besar nilai pembebanan maka semakin besar juga nilai K_i yang digunakan. Dan setelah mendapatkan *rule base* pada K_i maka yang dilakukan selanjutnya adalah mendapatkan *rule base* untuk nilai K_d. Berikut ini merupakan *rule base* untuk nilai K_d.

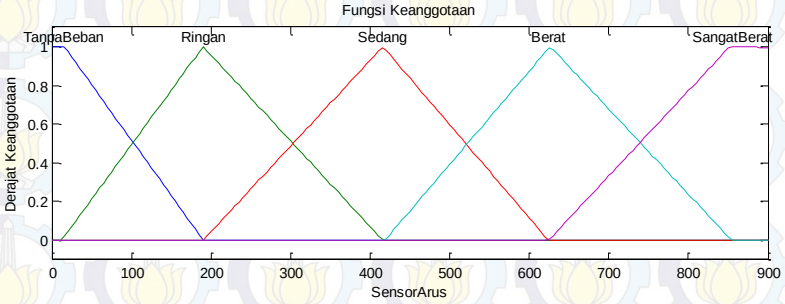
If X = 16 Then Y = 0,000
If X = 190 Then Y = 0,001
If X = 416 Then Y = 0,005
If X = 624 Then Y = 0,010
If X = 852 Then Y = 0,020

Pada rule base untuk penentuan nilai K_i di setiap pembebanan ini dapat dilihat bahwa semakin besar pembebanan maka semakin besar nilai K_i.

3.5.2.2 Fungsi Keanggotaan

Pada tahap awal harus ditentukan fungsi keanggotaan untuk *input* pada *fuzzy*. Karena sistem pengaturan motor BLDC V-1 ini merupakan *fuzzy* yang merubah nilai *gain* K_p, K_i, K_d pada setiap pembebanan, maka *input* fungsi keanggotaan harus berupa nilai yang merepresentasikan besarnya gaya rem yang dikeluarkan. Karena pada rem ini telah

didapatkan data identifikasi pada pembebanan tanpa beban, ringan, sedang, berat, dan sangat berat maka pada fungsi keanggotaan ini juga memiliki 5 anggota yaitu fungsi keanggotaan yang memiliki nilai tengah yang mewaikili beban 0%, lalu untuk beban ringan fungsi keanggotaan yang memiliki nilai tengah yang mewakili beban 20%, dan selanjutnya hingga beban sangat berat yang memiliki nilai tengah yang mewakili 80%. Pada sub judul *Software Arduino* telah dijelaskan bahwa pada serial yang dikirimkan ke Simulink MATLAB terdapat data rem yang memiliki *range* dari 1 sampai 999 yang mewakili *output* sensor arus. Oleh karena itu pada fungsi keanggotaan akan terdapat 5 anggota yang memiliki nilai tengah 16, 190, 416, 624, dan 852. Berikut fungsi keanggotaan dari fungsi keanggotaan *input.*



Gambar 3.12 Fungsi Keanggotaan *Input.*

Pada gambar di atas dapat dilihat bahwa jenis fungsi keanggotaan yang dipakai adalah jenis triangular. Jenis ini dipakai karena lebih mudah dalam perhitungannya. Pada fungsi keanggotaan di atas dapat dilihat bahwa setiap anggota memiliki jarak sekitar 400 dari kanan ke kiri. Pada Gambar 3.12 merupakan contoh dari fungsi keanggotaan dari nilai K_p. Dan untuk nilai K_i dan K_d memiliki struktur yang sama karena memiliki *input* yang sama yaitu rem. Perbedaan dari struktur *fuzzy* pada K_p, K_i, dan K_d berada pada fungsi keanggotaan *output*. Karena fungsi keanggotaan pada *output* memiliki nilai yang berbeda konstantanya dan nilai tersebut dapat dilihat pada Gambar 3.12.

Pada fungsi keanggotaan *output* terdapat 5 anggota pada setiap K_p, K_i, dan K_d. Bentuk dari fungsi keanggotaan pada hanya berupa konstanta yang merupakan nilai K_p, K_i, dan K_d. Pada perancangan ini

nilai K_p, K_i, dan K_d yang telah didapatkan hanya dibuat untuk dikalikan dengan nilai *weight* pada tiap *rule*-nya. Yang setelah itu hasil kali antara *weight* dengan niali K_p, K_i, dan K_d akan dijumlahkan dan akan dibagi dengan jumlah total *weight.*

BAB 4
PENGUJIAN DAN ANALISA

4.1 Gambaran Umum Pengujian Sistem

Pada tahapan ini akan dilakukan beberapa jenis pengujian yaitu pengujian sensor, pengujian *open loop* dari motor BLDC V-1, lalu pengujian kontroler yang disimulasikan pada hasil identifikasi model, lalu yang terakhir merupakan pengujian implementasi kontroler pada motor BLDC V-1.

Pengujian pertama merupakan pengujian sensor *rotary encoder* yang dilakukan dengan cara menbandingkan *output* yang dihasilkan oleh sensor *tachogenerator* dengan sensor *rotary encoder*. Pengujian kedua merupakan pengujian sistem *open loop* pada motor BLDC V-1, hal ini dilakukan dengan memberi *input* pada motor BLDC V-1 lalu dilihat *output* kecepatan yang terbaca oleh sensor *rotary encoder* dan dilihat hubungannya. Pengujian ketiga merupakan simulasi sistem, hal ini dillakukan dengan cara memasang kontroler yang telah dibuat pada hasil identifikasi model. Dan yang terakhir merupakan implementasi kontroler pada motor BLDC V-1.

4.2 Pengujian Sensor *Rotary Encoder*

Pada pengujian ini dilakukan pengecekan perbedaan antara kecepatan yang diukur oleh sensor *rotary encoder* dan program yang telah dibuat dengan sensor *tachogenerator* yang buatan pabrik. Setelah dilihat perbedaannya pada pengujian untuk pertama kalinya ternyata *sensor* dan program hasil memiliki *error* sekitar 200 rpm. Setelah program diperbaiki lebih lanjut didapatkanlah program yang menyesuaikan hasil pembacaan sensor dari *tachogenerator*.

Tabel 4.1 *Output* Pembacaan Kecepatan Motor dari 2 Sensor.

Input PWM	Pembacaan Sensor *Rotary Encoder* (rpm)	Pembacaan Sensor Laser *Tachometer* (rpm)
50	340,9	349
100	1099	1177
150	1722	1770
200	2196	2265

Tabel 4.1 Lanjutan.

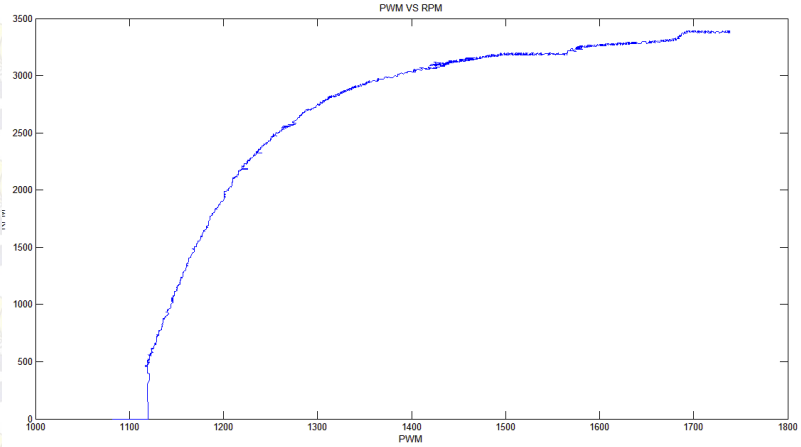
Input PWM	Pembacaan Sensor *Rotary Encoder* (rpm)	Pembacaan Sensor Laser *Tachometer* (rpm)
250	2503	2570
300	2751	2790
350	2905	2945
400	3043	3055
450	3115	3148
500	3199	3204
550	3236	3242
600	3300	3267
650	3319	3319
700	3363	3363
750	3370	3383
800	3409	3418
850	3411	3421
900	3471	3452
950	3512	3531
1000	3529	3538

Pada pengujian di atas dapat dilihat bahwa *output* hasil pembacaan pada sensor *rotary encoder* dan pada *tachogenerator* memiliki sedikit perbedaan hingga dapat mencapai 70 rpm. Perbedaan kecepatan ini semakin besar pada saat kecepatan tinggi dan besarnya getaran kecepatan yang dibaca juga semakin besar. Hal ini bisa disebabkan karena getaran dari poros motor yang sudah tidak lurus lagi yang menyebabkan getarnya piringan *encoder* saat di putar pada kecepatan tinggi. Oleh karena itu, pada pembacaan kecepatan piringan terjadi gangguan karena lubang yang melewati *encoder* akan berubah waktunya karena getaran tersebut.

4.3 Pengujian Pengujian *Open Loop* Kecepatan Motor

Pengujian ini dilakukan untuk melihat hubungan antara *input* dan *output* dari motor BLDC V-1. *input* pada motor BLDC V-1 yang

sebenarnya merupakan *output* yang diberikan oleh ESC yang berupa tegangan tiga fasa. Pada pengujian ini *input* yang dimaksud bukan *input* motor BLDC V-1 melainkan *input* dari ESC yang berupa sinyal PWM yang memiliki *duty cycle* dari sekitar 5% sampai 10%. Oleh karena itu, pengujian ini sebenarnya menguji *input* dan *output* dari motor BLDC V-1 dan ESC secara kesatuan. Lalu nilai dari PWM dari *range* tersebut akan di *mapping* menjadi nilai *throttle* dari 0 sampai 1000. Lalu *output* kecepatan memiliki satuan rpm yang berasal dari sensor *rotary encoder*. Pada pengujian ini diambil *sample* yang mencapai ribuan sehinnga data tidak dapat ditampilkan dengan menggunakan tabel. Oleh karena itu, data yang ditampilkan di sini hanya berupa grafik. Berikut grafik hasil pengujian *open loop* kecepatan motor.



Gambar 4.1 Grafik Hubungan *Input Output* Kecepatan Motor.

Dari grafik di atas dapat dilihat bahwa hubungan *input* dan *output* pada motor BLDC tidak linier. Setelah didapatkan hubungan garis antara *input* dan *output* maka hal yang dilakukan setelah itu adalah mencari persamaan garis dari hubungan tersebut. Dengan memisalkan sinyal *input throttle* dengan variabel x dan sinyal *output* kecepatan motor dengan variabel y. Didapatkanlah persamaan garis yang mendekati dengan hubungan *input output*. RMSE yang didapatkan dari model ini bernilai 23,6. Berikut persamaan garis yang memodelkan hubungan *input* dan *ouput* pada motor BLDC V-1.

$$y = p_1x^5 + p_2x^4 + p_3x^3 + p_4x^2 + p_5x + p_6 \quad (4.1)$$

Dengan nilai

$p_1 = 3{,}923e - 14$
$p_2 = -3{,}187e - 10$
$p_3 = 9{,}742e - 07$
$p_4 = -0{,}001356$
$p_5 = 0{,}9011$
$p_6 = -153{,}3$

Persamaan ini didapatkan dengan menggunakan *toolbox curve fitting* yang ada pada MATLAB.

4.4 Simulasi Sistem

Simulasi merupakan salah satu hal yang harus dilakukan sebelum mengimplementasikan kontroler pada *plant*. Dengan hasil simulasi yang sesuai akan mempermudah dalam penerapan kontroler agar sistem mencapai kriteria yang diinginkan oleh perancang.

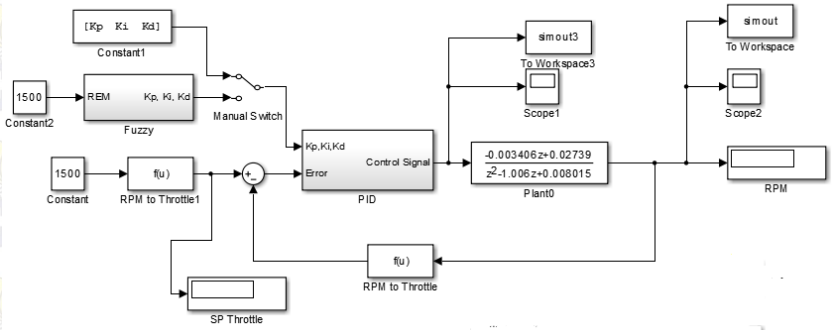
Sebelum melakukan simulasi dan implementasi perlu di tentukan dahulu waktu yang diperlukan untuk melakukan simulasi dan implementasi. Untuk simulasi dan implemntasi kali ini waktu yang dipakai untuk menjalankan sistem adalah 500 waktu *sampling* pada Simulink MATLAB. Waktu ini dapat diltentukan pada *toolbar* Simulink yang berada di samping tombol *running*. Lalu dilakukan pengujian seberapa lama waktu yang dibutuhkan untuk menjalankan Simulink tersebut dengan menggunakan *stopwatch*. Setelah dilakukan penghitungan dengan menggunakan *stopwatch* ternyata waktu yang dibutuhkan untuk menjalankan 500 waktu sampling dibutuhkan adalah 45,5 detik. Oleh karena itu, didapatkan sebuah konstatnta perkalian yang harus dikalikan pada hasil waktu yang ada pada *workspace* di MATLAB. Konstanta ini dapat dihitung sebagai berikut.

$$K = \frac{45.5}{500}$$
$$K = 0{,}091$$

Nilai ini nantinya akan dikalikan pada hasil workspace dari waktu simulasi dan implementasi. Oleh karena itu, setiap melakukan simulasi dan implementasi di semua pembebanan yang harus dilakukan adalah mengalikan konstanta ini dengan waktu simulasi dan implementasi.

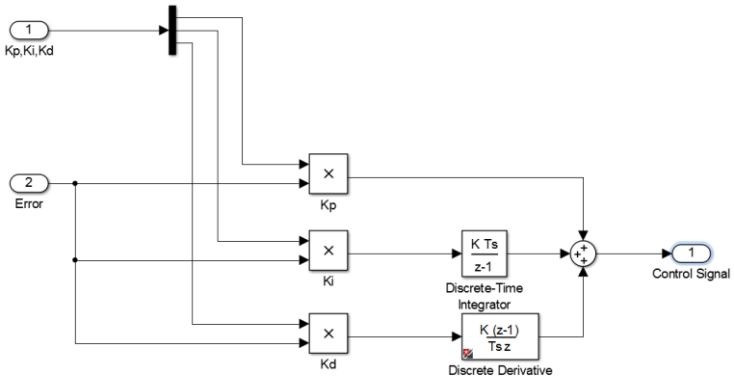
4.4.1 Diagram Blok Simulasi Sistem

Berikut ini merupakan blok untuk simulasi dengan menggunakan kontroler.



Gambar 4.2 Blok Simulink Simulasi Sistem.

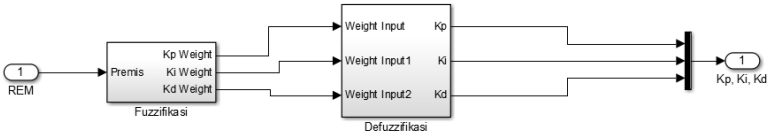
Bisa dilihat program MATLAB untuk simulasi ini berupa *close-loop* karena terdapat *summing point* yang membandingkan nilai *set point* dengan nilai pembacaan kecepatan. Pada sistem ini terdapa dua kontroler yaitu kontroler *fuzzy* dan kontroler PID. Berikut ini merupakan blok diagram untuk kontroler PID.



Gambar 4.3 Blok Simulink Kontroler PID.

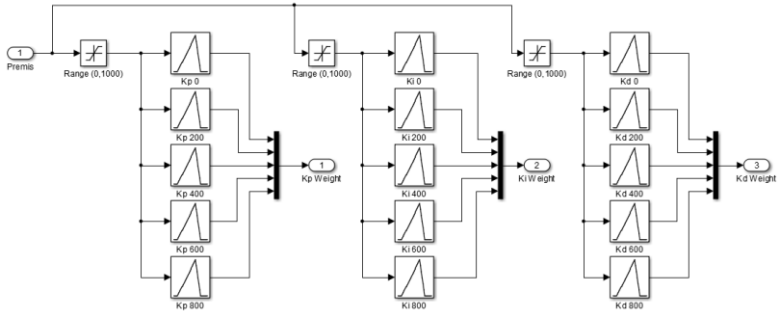
Dari gambar di atas bisa dilihat struktur dari PID dipisah antara kontroler P, kontroler I, dan kontroler D. Hal ini dipakai untuk memudahkan kontroler *fuzzy* untuk merubah parameter K_p, K_i, dan K_d. Sehinnga kontroler *fuzzy* hanya mengeluarkan nilai K_p, K_i, dan K_d yang tinggal dikalikan pada nilai *error* yang akan masuk ke kontroler P, kontroler I, dan kontroler D.

Lalu berikut ini merupakan blok diagram untuk kontroler *fuzzy*, pada *fuzzy* blok diagram dibagi menjadi dua tahapan yaitu tahapan unttuk mencari weight atau fuzzifikasi dan yang kedua tahapan untuk defuzzifikasi.



Gambar 4.4 Blok Simulink *Fuzzy.*

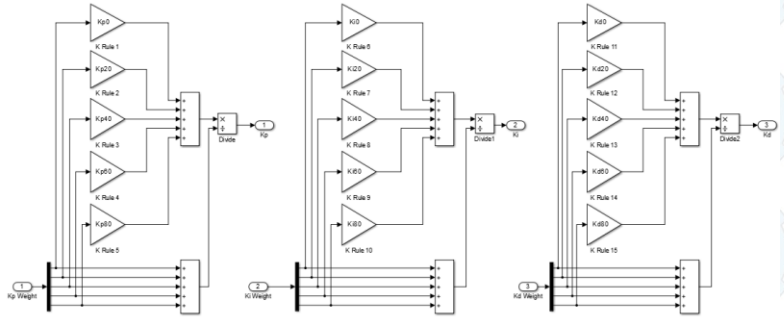
Seperti yang kita sudah ditulis sebelumnya *fuzzy* memliki 4 elemen yaitu *fuzzy rule base*, *fuzzy interface engine*, fuzzifikasi, defuzzifikasi. Pada blok Simulink di atas merupakan blok fuzzifikasi dan blok defuzzifikasi. Pada gambar berikutnya merupakan blok Simulink untuk fuzzifikasi dan defuzzifikasi.



Gambar 4.5 Blok Simulink Fuzzifikasi.

Bisa dilihat bahwa terdapat tiga fungsi keanggotaan yang masing-masing mewakili nilai K_p, K_i, dan K_d. Pada setiap nilai K_p, K_i, dan K_d

terdapat lima anggota yang merupakan fungsi keanggotaan untuk *input* rem. 5 rem tersebut dibagi menjadi merupakan *input* rem tanpa beban, beban ringan, sedang, berat, sangat beban yang diwakili dengan fungsi keanggotaan yang berbentuk triangular yang memiliki nilai tengah 16, 190, 416, 624, dan 852. Dari keluaran subsistem ini akan dihasilkan lima nilai *weight* untuk setiap *input* di setiap nilai K_p, K_i, dan K_d. Oleh karena itu, *output* dari subsistem ini terdapat 15 *weight*. Setelah itu 15 nilai *weight* ini akan diolah pada subsistem defuzzifikasi. Berikut blok Simulink untuk subsistem defuzzifikasi.



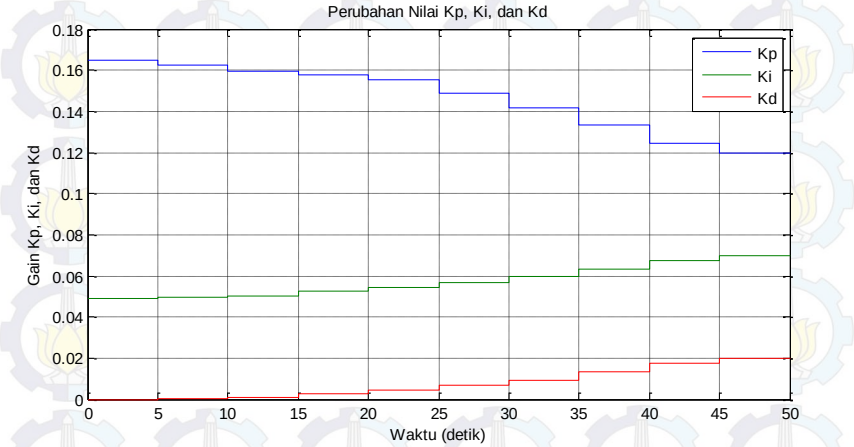
Gambar 4.6 Blok Simulink Defuzzifikasi.

15 buah nilai *weight* pada subsistem ini akan dikalikan dengan fungsi keanggotaan *input* yang sebelumnya telah dihitung pada subbab perancangan kontroler. Lalu nilai *weight* yang telah dikalikan dengan fungsi keanggotaan *output* akan dijumlahkan semuanya dan akan dibagi dengan nilai *weight* yang telah dijumlahkan pula. Oleh karena itu, *output* dari sistem ini merupakan nilai K_p, K_i, dan K_d. yang telah ditentukan pada subbab perancangan kontroler. Nilai K_p, K_i, dan K_d disimpan pada blok *gain* yang terhubung dengan *workspace*. Oleh karena itu, untuk bisa menjalankan program Simulink MATLAB untuk simulasi ini harus dijalankan sebuah *file* MATLAB yang menyimpan nilai K_p, K_i, dan K_d di setiap pembebanan.

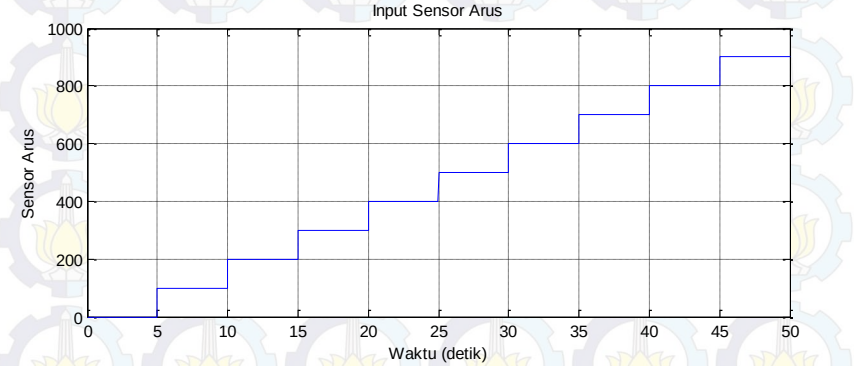
4.4.2 Pengujian Perubahan Nilai K_p, K_i, Dan K_d

Pada tahapan ini aka diuji bagaimana nilai K_p, K_i, dan K_d berubah sesuai dengan perubahan dari nilai pembacaan sensor arus. Pada pengujian ini akan dilihat *output* dari *gain scheduler fuzzy* yang

memiliki nilai K_p, K_i, dan K_d. Berikut merupakan hasil pengujian dari *gain scheduler fuzzy*.



Gambar 4.7 Grafik Perubahan Nilai K_p, K_i, dan K_d.



Gambar 4.8 *Input* Uji Perubahan Nilai K_p, K_i, dan K_d.

Dari dua gambar di atas dapat dilihat bagaimana perubahan nilai K_p, K_i, dan K_d ketika nilai sensor arus naik secara perlahan. Dapat dilihat ketika terjadi kenaikan nilai sensor arus maka nilai K_i dan K_d akan naik pula sedangkan untuk nilai K_p menurun. Dan dari grafik di atas dapat diambil data sebagai berikut.

Tabel 4.2 Tabel Perubahan Nilai K_p, K_I, dan K_d.

Pembacaan Sensor Arus (818,4 x Volt)	K_p	K_i	K_d
0	0,16500	0,049000	0,000000
100	0,16253	0,049494	0,000494
200	0,15978	0,050221	0,001177
300	0,15757	0,052434	0,002947
400	0,15535	0,054646	0,004717
500	0,14894	0,057019	0,007019
600	0,14173	0,059423	0,009423
700	0,13333	0,063333	0,013333
800	0,12456	0,067719	0,017719
900	0,12000	0,070000	0,020000

4.4.3 Pengujian Respons Dengan Kontroler

Pada tahapan ini kontroler yang telah dibuat akan dicoba dijalankan pada model hasil identifikasi sistem. Pengujian ini dilakukan dengan memberikan respon *step* di semua pembebanan mulai dari tanpa beban hingga beban sangat berat. Setelah itu respon akan dianalisa untuk dicari nilai *rise time*, *settling time*, *overshoot*, dan *error steady state*.

Pengujian dilakukan dengan memberikan sinyal *step* pada semua fungsi alih di setiap pembebanan. Sinyal *step* yang diberikan merupakan *throttle* sebesar 147 yang seharusnya akan menghasilkan *output* kecepatan motor sebesar kisaran 1750 rpm.

Setalah dilakukan simulasi akan dilakukan analisa terhadap respon hasil dari kontroler. Yang dilakukan selanjutnya adalah menghitung *settling time*, *rise time*, *overshoot* dan *error steady state*. Namun karena kontroler yang digunakan adalah kontroler PID dan sifat dari kontroler yang memiliki integrator adalah menghilangkan nilai *error steady state*, sehingga yang perlu dihiutung pada analisa kali ini hanya nilai *settling time*, *rise time*, dan *overshoot.*

Untuk menghitung nilai *settling time* yang perlu dilakukan hanyalah menghitung nilai di mana respon *output* telah memasuki zona ±5% atau ±2% atau ±0,5% dari nilai *steady state* respon. Lalu untuk *rise*

time yang perlu dilakukan menghitung waktu yang dibutuhkan oleh respon dari mulai 10% dari nilai *steady state* hingga ke 90% dari nilai *steady state.* Dan untuk nilai *overshoot* yang perlu dilakukan adalah menghitung nilai nilai maksimum respon lalu dikurangi nilai *steady state* respon lalu dibagi dengan nilai *steady state* respon.

4.4.3.1 Simulasi Pada Kondisi Tanpa Beban

Gambar 4.9 Respon *Step* pada Kondisi Tanpa Beban.

Pada respon motor BLDC tanpa beban tidak terdapat *error steady state*. Begitu juga nilai *overshoot* karena nilai maksimum pada respon merupakan nilai *seatdy state* pada respon. Dan di bawah ini merupakan hasil perhitungan dan pengamatan *rise time* (T_r) dan *setlling time* (T_s) dari respon sistem.

Kecepatan *steady state* motor adalah 1752 rpm, sehingga untuk mencari T_s yang perlu dicari adalah waktu dari 0 hingga titik di mana kecepatan motor mencapai 95% dari kecepatan maksimum yang nilainya yaitu 1664,4 rpm.

Dan untuk mencari nilai T_r yang perlu dicari adalah waktu di mana kecepatan motor mencapai 90% dari kecepatan maksimum dikurangi dengan waktu di mana kecepatan motor mencapai 10% dari kecepatan maksimum. Di mana nilai kecepatan motor saat 10% dari kecepatan maksimum adalah 1576,8 rpm dan kecepatan motor saat 90% dari kecepatan maksimum adalah 175,2 rpm. Oleh karena itu, dari nilai-nilai ini hanya perlu dicari waktu pada saat titik-titik nilai tersebut.

$T_s = 4{,}42\ detik$
$T_r = 3{,}33\ detik$

Yang dilakukan setelah tahap ini adalah pengujian saat beban ringan, sedang, berat, sangat berat. Lalu pada pengujaian tersebut dilakukan langkah yang sama yaitu menghitung nilai *rise time*, *settling time*, dan *overshoot*.

4.4.3.2 Analisa Hasil Pengujian

Setelah semua fungsi alih disimulasikan dan semua data mengenai *rise time*, *settling time*, *overshoot*, dan *error steady state* didapatkan maka berikut ini hasil rangkuman data yang didapatkan.

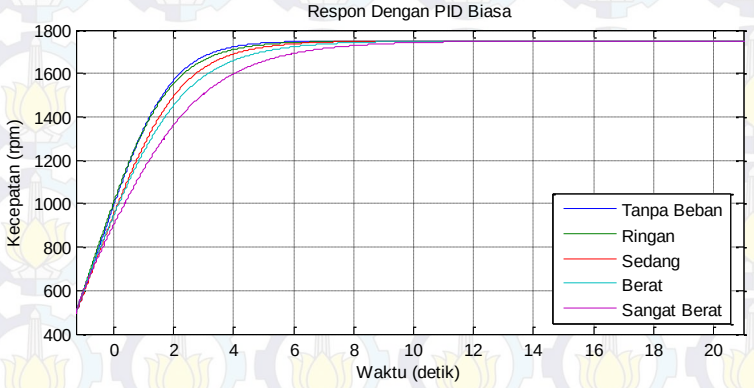
Tabel 4.3 Data Karakteristik Respon Hasil Simulasi.

Pembebanan	*Settling Time* (detik)	*Rise Time* (detik)	*Overshoot*	*Error Steady State*
Tanpa Beban	4,42	3,33	0	0
Ringan	4,58	3,44	0	0
Sedang	4,53	3,40	0	0
Berat	4,62	3,46	0	0
Sangat Berat	4,72	3,53	0	0

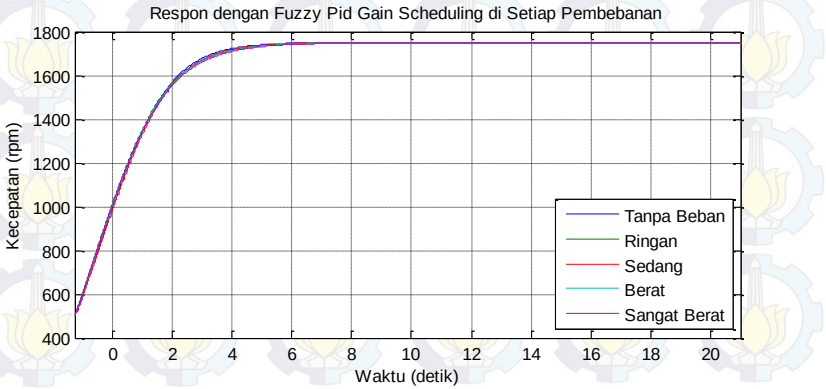
Untuk nilai *overshoot* dan *error steady state* tidak dicantumkan karena keduanya memiliki nilai nol di semua pembebanan.

Dari data ini dapat dilihat bahwa rata-rata nilai *settling time* berkisar pada waktu 4,5 detik dan rata-rata nilai *rise time* berkisar pada 3,5 detik. Pada *steady state* sinyal kontrol dapat terlihat bahwa semakin naik nilai pembebanan maka semakin naik juga nilai sinyal kontrol yang dibutuhkan. Hal ini terjadi karena untuk me+mutar motor pada beban yang lebih besar di kecepatan yang sama dibutuhkan *input* yang lebih besar sehingga *input* ini dapat mengkompensasi kekurangan kecepatan pada motor.

Untuk pengaturan kecepatan motor dengan menggunakan kontroler PID saja sebenarnya sudah cukup. Tetapi, tujuan dari kontroler ini adalah membuat respon kecepatan motor memiliki nilai *settling time* yang sama pada setiap kondisi beban maka harus terlihat perbedaannya ketika sistem menggunakan kontroler PID saja dan kontroler menggunakan kontroler *Fuzzy* PID *Gain Scheduling*. Berikut ini merupakan grafik yang membandingkan nilai *settling time* di setiap kondisi beban dengan kontroler PID dan dengan kontroler *Fuzzy* PID *Gain Scheduling*.



Gambar 4.10 Respon dengan Kontroler PID.



Gambar 4.11 Respon dengan Kontroler *Fuzzy* PID *Gain Scheduling*.

Berikut ini merupakan tabel yang membandingkan nilai *settling time* di setiap kondisi beban dengan kontroler PID dan dengan kontroler *Fuzzy* PID *Gain Scheduling*.

Tabel 4.4 Tabel Perbandingan *Settling Time.*

Pembebanan	*Settling Time PID* (detik)	*Setting Time PID Fuzzy* (detik)
Tanpa Beban	4,42	4,42
Ringan	4,70	4,58
Sedang	5,22	4,53
Berat	5,87	4,62
Sangat Berat	7,08	4,72

4.5 Implementasi Sistem

Pada tahapan ini kontroler yang telah dibuat akan dicoba dijalankan pada model hasil identifikasi sistem. Pengujian ini dilakukan dengan memberikan respon *step* di semua pembebanan mulai dari kondisi tanpa beban hingga beban sangat berat.

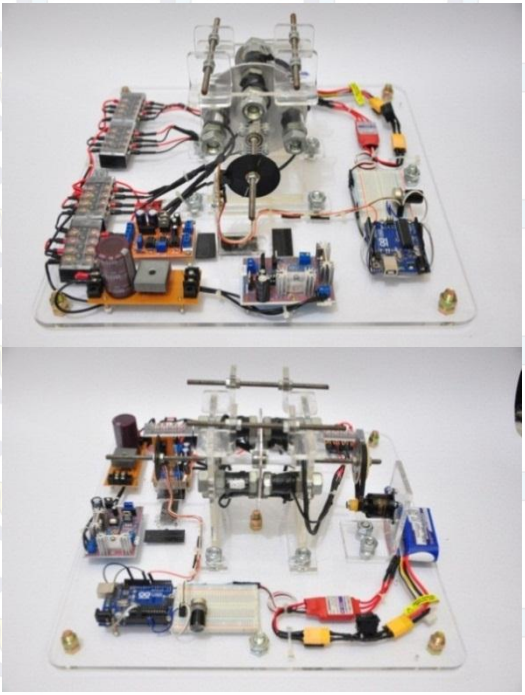
4.5.1 Realisasi *Plant*

Berikut ini merupakan *plant* dari sistem pengaturan motor BLDC V-1 yang telah di realisasikan. *Plant* ini memiiliki dimensi sekitar 40x40x15 cm. *Plant* ini terbagi menjadi 3 sistem. Yaitu sistem yang bekerja menjalankan motor, lalu sistem yang menjalankan rem, dan satu lagi sistem yang menjalankan sensor. Dari 3 sistem ini untuk bisa terhubung dengan kontroler harus memiliki *interface*. *Interface* yang kami gunakan adalah Arduino Uno.

4.5.1.1 Realisasi Fisik Plant

Berikut ini sebagian gambar yang diambil dari *plant* yang telah direalisasikan. Pada *plant* ini sebagian besar bahan dasar yang dipakai untuk dudukan terbuat dari akrilik. Pada *plant* ini terdapat 4 buah PCB

yang diantaranya rangkaian untuk sensor *rotary encoder*, lalu rangkaian penyearah sinyal AC atau *rectifier* atau kadang disebut *voltage regulator*, lalu terdapat rangkaian *driver* rem magnetik yang berfungsi sebagai pengatur tegangan *output* atau daya yang dekeluarkan oleh rem magnetik., lalu yang terakhir terdapat rangkaian sensor arus yang berfungsi membaca nilai arus pada lilitan rem magnetik. Setelah itu terdapat 8 buah lilitan yang berfungsi sebagai rem magnetik, lalu terdapat 4 buah terminal yang menghubungkan rem magnetik dengan *recctifier*. Selain itu terdapat 2 buah piringan yang satu terbuat dari alumunium yang berfungsi sebagai beban pada poros dan yang satu terbuat dari besi yang berfungsi sebagai *rotary encoder*. Lalu terdapat ESC dan motor BLDC V-1. Setelah itu terdapat juga Arduino sebagai *interface* antara *plant* dengan komputer. Setelah itu terdapat juga *breadboard* yang menghubungkan *plant* dengan Arduino.

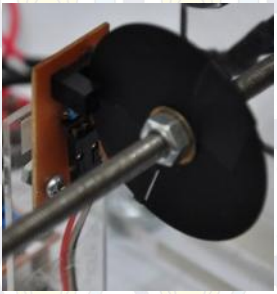


Gambar 4.12 *Plant* Motor BLDC.

Plant motor BLDC V-1 ini di letakkan di atas *base* yang terbuat dari akrilik. Rem magnetik, sensor *rotary encoder* dan motor BLDC juga dipasangkan pada akrilik.

Gambar 4.13 Rem Magnetik.

Kontruksi rem magnetik adalah baut yang memiliki diameter 1,4 cm. lalu baut tersebut dililit sebanyak 283 lilitan agar didpatkan medan magnet yang cukup kuat untuk membebani motor BLDC V-1. Pada gambar tersebut juga terlihat ada piringan diantara dua baut yang terbuat dari alumunium setebal 2,5 mm. diameter dari pringan ini adalah 10 cm. piringan terpasang pada sebuah poros yang terpasang dengan *gear* yang terhubung dengan motor. Poros ini juga ikut menyebabkan piringan untuk sensor *rotary encoder* berputar



Gambar 4.14 Sensor *Rotary Encoder*.

Ini merupakan piringan *encoder* yang terbuat dari besi yang berdiameter 5 cm. pringan ini di warnai dengan warna hitam untuk mengurangi gangguan pantulan cahaya dari luar yang dapat menyebabkan *phototransistor* terganggu. Pada piringan ini diberi 2 buah lubang yang memiliki lebar 1 mm. dipilih 2 lubang karena *processor* pada Arduino dikhawatirkan tidak dapat membaca kecepatan dari *encoder* apabla piringan dibuat lebih dari 2 lubang. Setelah itu *encoder* ini akan ditutup semuanya dengan sebuah karton untuk menambah keandalan dari *encoder*.

4.5.1.2 Program Arduino

Pada dasarnya pada program Arduino hanya berfungsi sebagai pengirim data dan penerima data dari komputer. Pada dasarnya program Arduino yang dibuat kali ini memiliki 2 sistem utama yaitu inisialisasi dan program utama. Berikut ini program inisialisasi yang digunakan saat implementasi.

```
#include <Servo.h>
Servo esc;
int setpoint;
int incomingByte;
int percenttrot;
int encoderPin = 7;
unsigned long duration;
unsigned long rpm;
char disp2[1];
char disp3[1];

void setup() {
  Serial.begin(115200);
  pinMode(encoderPin, INPUT);
  esc.attach(9);
  esc.writeMicroseconds(1080);
  delay(2000);
```

Gambar 4.15 Potongan Program Inisialisasi.

Pada inisialisasi terdapat beberapa hal yang dilakukan yaitu menyebutkan jenis-jenis *library* yang dipakai, lalu menyebutkan variabel-variabel yang dipakai, lalu menyebutkan nilai *baudrate*, dan yang terkahir menyebutkan pin-pin yang dipakai pada Arduino.

Pada program di atas dapat dilihat bahwa untuk implementasi dibutuhkan *library* servo, *library* ini berfungsi untuk menjalankan motor sehingga pada motor hanya perlu dimasukkan nilai-nilai tertentu dan motor akan dapat langsung berputar. Lalu terdapat bebrapa jenis variabel yang dipakai diantaranya *integer, character*. dan *unsigned long*. Lalu *baudrate* yang dipakai bernilai 115200. Dan pin yang digunakan pada *plant* adalah pin 7 sebagai *encoder* dan pin 9 terhubung dengan ESC.

Pada program utama Arduino akan terjadi program yang terus menerus dilakukan selama Arduino dalam kondisi menyala. Pada program utama terbagi menjadi 3 buah sistem yang terjadi yang pertama merupakan program yang membaca angka yang dikirim dari Simulink MATLAB, lalu yang kedua merupakan program untuk membaca nilai kecepatan motor dalam satuan rpm, dan yang ketiga merupakan program untuk *driver* pada rem magnetik. Berikut ini merupakan program untuk menerima angka dari komputer.

```
void loop() {
  setpoint = map(percenttrot,0,1000,1080,1750);
  esc.writeMicroseconds(setpoint);
  char displ[10];
  if((setpoint<=1751)&&(setpoint>=1079)){
    sprintf(displ,"*%4d",percenttrot);
    Serial.print(displ);
  }
  else{
    setpoint=1080;
    percenttrot=0;
    sprintf(displ,"*%4d",percenttrot);
    Serial.print(displ);
  }
  if (Serial.available()>0){
    percenttrot = Serial.parseInt();
```

Gambar 4.16 Potongan Program Membaca Serial.

Pada program ini digunakan fungsi Serial.ParseInt( ) yang berfungsi untuk membaca angka yang diberikan pada serial monitor pada Arduino. Fungsi ini hanya dapat membaca angka yang masuk pada

serial monitor sehingga apabila pada serial monitor terdapat variabel yang bukan angka akan dihiraukan dan dilewati pembacaanya.

Lalu program ini akan menuliskan hasil pembacaannya pada serial monitor dengan menggunakan Serial.print( ).

Lalu berikut ini merupakan program untuk membaca kecepatan motor BLDC dalam satuan rpm.

```
duration = pulseIn(encoderPin, HIGH, 100000);
rpm=(79550000/3)/duration;
if (rpm>4000){
  rpm=0;}
Serial.print('#');
sprintf(disp2,"%4d",rpm);
Serial.print(disp2);
```

Gambar 4.17 Potongan Program Sensor Kecepatan.

Pada bagian ini program akan membaca nilai waktu yang dibutuhkan untuk tiap *pulse*. Lalu waktu ini akan dikonversi dalam satuan rpm. Setelah itu dilakukan program ini akan menuliskan nilai kecepatan motor tersebut pada serial monitor Arduino.

Lalu berikut ini merupapakn program untuk *driver* rem magnetik. Seperti yang telah dijelaskan pada bab sebelumnya rem magnetic ini dikendalikan kekuatannya dengan menggunakan sinyal PWM. Sinyal PWM ini merupakan sinyal yang berasal dari Arduino.

```
int sensorValue = analogRead(A0);
int rem = map(sensorValue,0,1023,1,254);
analogWrite(11,rem);
int percentrem = map(sensorValue,0,1023,1,999);
int sensorarus = analogRead(A2)*4;
sprintf(disp3,"%3d$%3d",percentrem,sensorarus);
Serial.print('@');
Serial.print(disp3);
```

Gambar 4.18 Potongan Program Untuk *Input* Rem dan Sensor Arus.

Sinyal PWM ini akan diatur lebarnya dengan menggunakan potensiometer. Oleh karena itu pada program ini dilakukan pembacaan pada potensiometer yang selanjutnya nilai tersebut dikonversikan dalam

bentuk sinyal PWM yang memiliki nilai 0 sampai 255 pada Arduino. Lalu setelah itu nilai akan di-*mapping* dengan nilai 1 sampai 999 yang selanjutnya aka ditulis pada serial monitor Arduino.

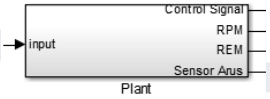
Pada serial monitor Arduino akan tertulis sebuah barisan angkan yang tertulis dalam format "*%4d#%4d@%3d$%3d" yang merupakan barisan satuan dari nilai-nilai data yang mau dikirimkan ke komputer. Format ini harus disesuaikan pada program Simulink MATLAB sehingga data yang diterima tidak kacau balau.

Format "*%4d#%4d@%3d" ini memiliki arti bahwa pada setiap baris pengiriman yang berisi format ini terdapat tiga data. Tiga data ini diantaranya adalah data yang dikirim oleh komputer ke Arduino, lalu yang kedua data sensor *rotary encoder*, dan yang ketiga dan keempat merupakan data pembacaan input rem dan sensor arus rem magnetik.

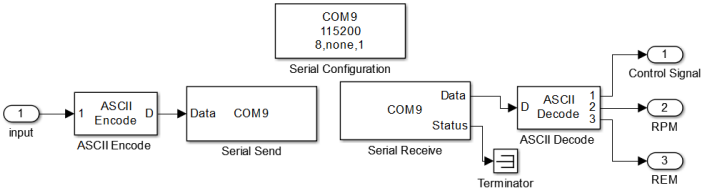
Format "*%4d#%4d@%3d$%3d" merupakan tiga buah data yang dibatasi dengan nilai "*", "#", dan "@". Lalu nilai %4d, dan %3d memiliki arti bahwa angka yang dikirim merupakan 4 digit apabila %4d dan 3 digit apabila %3d.

4.5.2 Blok Simulink Implementasi Sistem

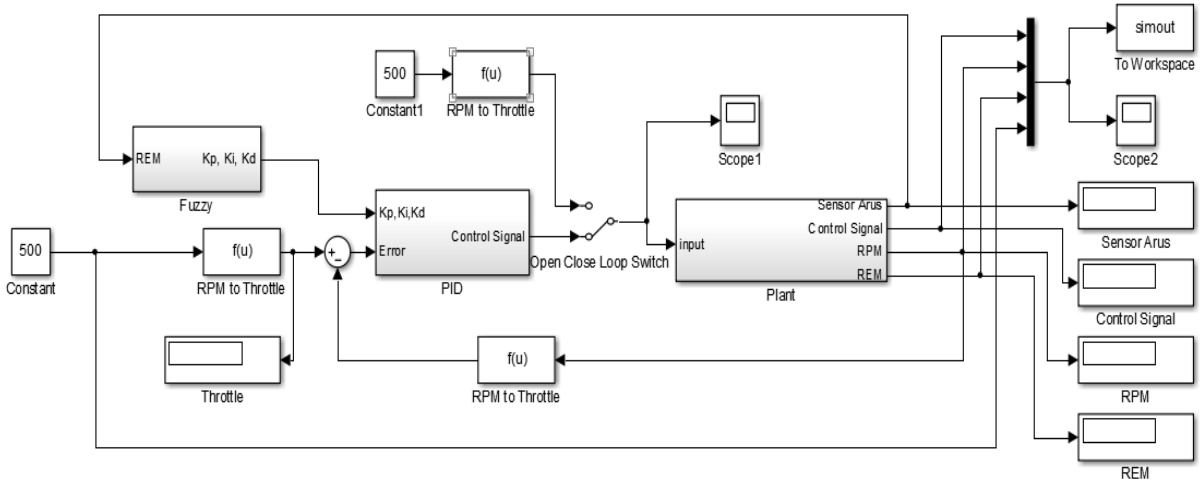
Pada dasarnya blok Simulink pada simulasi dan implementasi memiliki struktur yang sama, perbedaannya hanya di bagian blok pada *plant* saja. Jika pada simulasi sistem, Blok diagram pada *plant* diisi dengan fungsi alih yang telah didapatkan dari identifikasi. Tetapi pada implementasi sistem, blok dari plant pada *plant* berisi blok komunikasi serial.



Gambar 4.19 Blok *Plant* untuk Implementasi.



Gambar 4.20 Blok Simulink Untuk Komunikasi Serial.

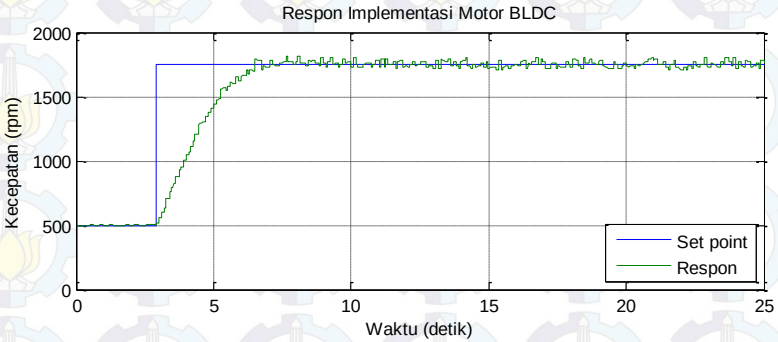


Gambar 4.21 Blok Simulink Implementasi Sistem.

4.5.3 Pengujian Respon Motor BLDC V-1 dengan Kontroler

Setelah dilakukan simulasi hal yang dilakukan selanjutnya adalah implementasi kontroler pada *plant real* motor BLDC. Pengujian ini dilakukan dengan memberikan *set point* kecepatan 500 rpm yang selanjutnya dinaikkan lagi *set point* menjadi 1750 rpm. Setelah itu respon dari motor dapat dianalisa dan dicari nilai dari *rise time*, *settling time*, *overshoot*, dan *error steady state*.

4.5.3.1 Implementasi Pada Kondisi Tanpa Beban

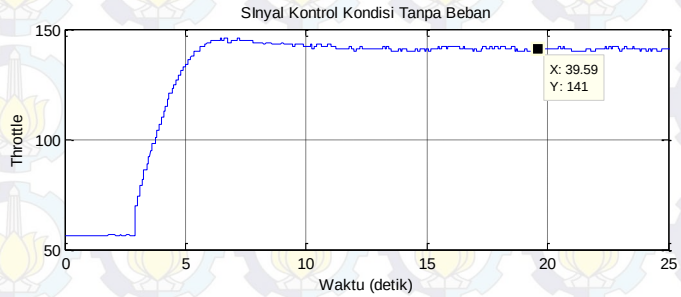


Gambar 4.22 Respon *Step* Implementasi pada Kondisi Tanpa Beban.

Pada saat *start up* dan motor diberi kontroler sebenarnya sinyal kontrol langsung melesat ke angka 130. Dan dari analisa penulis hal ini terjadi karena pada program *built in* pada ESC terdapat sedikit *delay* pada saat *start up*. Hal ini menjadi pertimbangan penulis karena pada saat dilakukan simulasi sinyal kontrol sama sekali sinyal repson yang melebihi sinyal kontrol saat *steady state.* Oleh karena itu kontroler ini hanya dapat terlihat responnya ketika motor sudah berputar pada kecepatan tertentu baru *set point* dinaikkan ke kecepatan yang lebih tinggi. Seperti yang terlihat pada Gambar 4.22 penulis membuat dua buah *input step*. *Input step* yang pertama berfungsi untuk *start up* yang memiliki *set point* 500 dan yang kedua merupakan *input step* yang ditambahkan 1250 rpm sehingga *set point* menjadi 1750 rpm. Maka dari itu untuk analisa kali ini yang dihitung merupakan respon pada *step* kedua yang merupakan perubahan *set point* dari 500 rpm menjadi *set point* 1750 rpm

Pada respon motor BLDC V-1 tanpa beban tidak terdapat *error steady state*. Namun berbeda dengan simulasi, pada implementasi terdapat *overshoot* karena nilai maksimum terdapat di atas nilai *steady state*. Dan di bawah ini merupakan hasil perhitungan dan pengamatan *rise time* (T_r), *setlling time* (T_s), dan *Percentage Overshoot* (%Mp) dari respon sistem. Pada kali ini analisa dilakukan pada saat kecepatan motor berubah dari 500 rpm menjadi 1750 rpm.

$T_s = 6{,}53 - 2{,}93$
$T_s = 3{,}6\ detik$
$T_r = 5{,}83 - 3{,}15$
$T_r = 2{,}68\ detik$
$\%Mp = (1766 - 1750)/1250$
$\%Mp = \ 0{,}0128 = 1{,}28\ \%$



Gambar 4.23 Sinyal Kontrol Implementasi pada Kondisi Tanpa Beban.

Bisa dilihat pada Gambar 4.23 bahwa sinyal kontrol pada saat motor BLDC dinyalakan terjadi kenaikan yang sangat drastis. Hal ini menyebabkan tingginya *overshoot* pada respon kecepatan motor BLDC V-1 pada saat *start up*. Seperti yang telah dijelaskan sebelumnya hal ini disebabkan oleh *start up* pada program *built in* ESC. Pada ESC yang digunakan pada ESC terdapat 3 jenis *start up* yaitu ada *start up* normal, pelan, dan sangat pelan. Pada tugas akhir ini digunakan *start up* dengan aturan normal dan ini merupakan *default* dari ESC. Dan hipotesa lain dari penyebab respon saat *start up* ini adalah pada hubungan *input* dan *output* motor BLDC. dari hasil pengujian *open loop* didapatkan bahwa motor tidak aka berputar sebelum *throttle* mencapai nilai 50.

Kembali ke pengujian respon bisa dilihat out *steady state* dari sinyal kontrol berada pada *throttle* 141. Hal ini berbeda dengan simulasi

yang *throttle*-nya bernilai 147. Hal ini dapat disebabkan banyaknya gangguan faktor luar seperti baterai, tegangnya *belt*.

Selain itu hal ini juga bisa disebabkan karena pada saat identifikasi, nilai yang diberikan juga bukan dari 0 sehinga kontroler *Fuzzy* PID ini hanya berlaku untuk kecepatan dari 500 sampai 1750. Maka dari itu pada pengujian kali ini respon yang diteliti merupakan respon dari motor BLDC dari kecepatan 500 ke 1750.

4.5.3.2 Analisa Hasil Implementasi

Sebelumnya telah ditampilkan penghitungan nilai *rise time*, *settling time*, dan *overshoot* pada saat kondisi motor tidak diberi beban. Setelah yang dilakukan adalah pengujian pada saat beban ringan, sedang, berat, dan sangat berat. Data-data itu dilakukan analisa dengan cara yang sama dengan analisa saat kondisi tanpa beban.

Setelah semua data telah diambil, berikut hasil seluruh data yang telah didapatkan dari mulai kondisi tanpa beban hingga saat kondisi beban sangat berat dalam bentuk tabel.

Tabel 4.5 Data Karakteristik Respon Hasil Implementasi.

Pembebanan	*Settling Time* (detik)	*Rise Time* (detik)	*Overshoot* (%)	*Error Steady State*
Tanpa Beban	3,60	2,80	1,28	0
Ringan	3,21	2,62	2,80	0
Sedang	3,13	2,44	1,20	0
Berat	3,69	2,67	0,56	0
Sangat Berat	3,77	2,88	0,24	0

4.5.4 Perbandingan Hasil Simulasi Dan Implementasi

Setelah dua jenis pengujian dilakukan di dua jenis pengujian yaitu simulasi dan implementasi, kedua data hasil pengujian dibandingkan untuk mengetahui bagaimana perbedaannya perhitungan hasil simulasi dan implementasi. Hal ini dilakukan dengan membandingkan nilai *rise time*, *settling time*, *overshoot*, dan *error steady state*. Perbandingan ini akan memperlihatkan bagaimana perbedaan respon kontroler yang

dipasang pada pemodelan motor BLDC V-1 dengan respon kontroler yang langsung dipasang pada motor BLDC V-1.

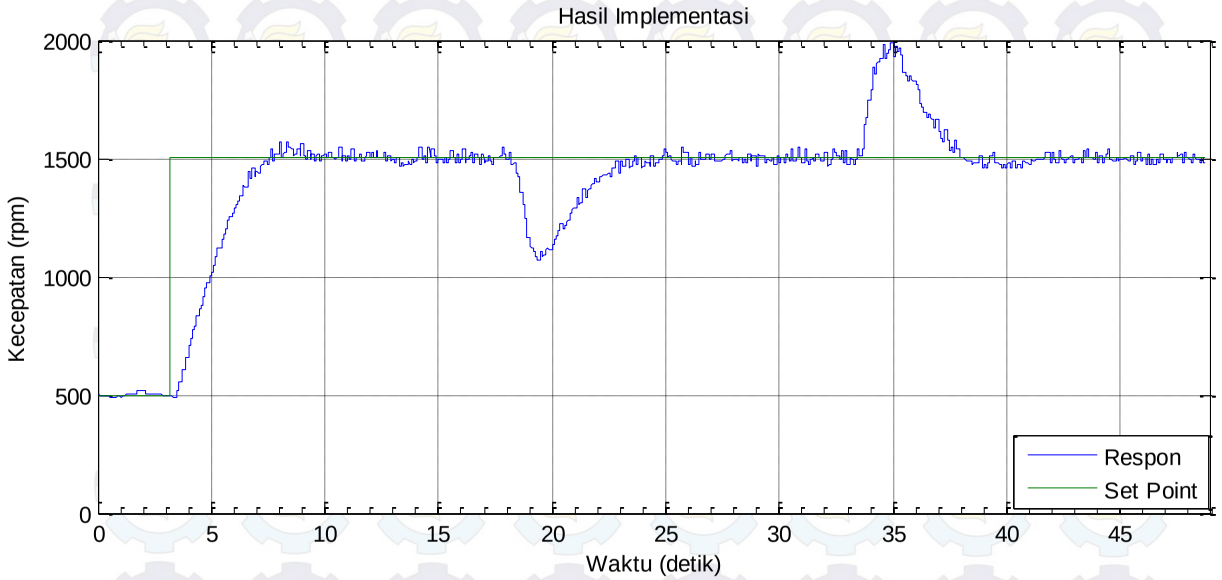
Berikut ini merupakan tabel perbandingan data antara hasil simulasi dengan implemntasi.

Tabel 4.6 Data Hasil Simulasi Dan Implementasi.

Pembebanan	*Settling Time* (detik)		*Rise Time* (detik)		*Overshoot*		*Error Steady State*	
	Sim.	Imp.	Sim.	Imp.	Sim.	Imp.	Sim.	Imp.
Tanpa Beban	4,42	3,60	3,33	2,80	0	1,28	0	0
Ringan	4,58	3,21	3,44	2,62	0	2,80	0	0
Sedang	4,53	3,13	3,4	2,44	0	1,20	0	0
Berat	4,62	3,69	3,46	2,67	0	0,56	0	0
Sangat Berat	4,72	3,77	3,53	2,88	0	0,24	0	0

Dari Tabel 4.6 bisa dilihat perbedaan nilai hasil simulasi dengan hasil implementasi. Pada hasil simulasi nilai *settling time* rata-rata berada pada nilai 4,5 detik sedangkan pada saat implementasi nilai *settling time* rata-rata 3,6 detik. Begitu juga dengan nilai *rise time*, pada simulasi nilai *rise time* rata-rata adalah 3,4 detik sedangkan saat implementasi rata-rata *rise time* adalah 2,6 detik. Lalu pada saat simulasi, respon simulasi tidak memiliki *overshoot* sedangkan respon saat implementasi memiliki sedikit *overshoot* walaupun relatif kecil yaitu sekitar 1 persen. Hal ini bisa terjadi karena pemodelan dari motor BLDC yang berbeda sedikit dari *plant* motor BLDC V-1.

Dan gambar di halaman berikut ini merupakan gambar hasil implementasi kontroler *Fuzzy* PID *Gain Scheduling* yang pada *input*-nya diberi *set point* 500 rpm untuk *start up* dan lalu diberi nilai 1500 rpm. Setelah itu nilai rem pun dimainkan dan berikut respon hasil *running* dari *plant* motor BLDC V-1.



Gambar 4.24 Hasil Implementasi.

Pada Gambar 4.24 merupakan hasil implementasi kontroler dengan beban yang berubah-ubah. Hal ini dilakukan dengan cara menjalankan motor pada kecepatan tertentu lalu beban pada rem dinaikkan hingga ke tingkat beban sangat berat yaitu pembacaan sekitar 852. Setelah itu rem kembali dihilangkan kembali sehingga motor kembali ke *plant* semula.

Setelah dilakukan hal tersebut dapat dilihat hasil respon dari kecepatan motor setelah respon diubah-ubah. Pada grafik kecepatan vs waktu di atas bisa dilihat respon kontroler saat motor diberikan beban sebesar 852 pada detik ke 17. Kecepatan pada motor langsung turun sampai sekitar 1100 rpm, lalu kontroler pun langsung memberikan respon untuk mengembalikan kecepatan motor agar sesuai dengan *set point*. Begitu juga ketika rem dikembalikan pada posisi tanpa beban, nilai kecepatan motor naik ketika beban dilepas setelah itu kontroler akan mengembalikan kecepatan motor agar berada kembali pada keadaan *set point*.

BAB 5
PENUTUP

5.1 Kesimpulan

Setelah dilakukan pengujian dan analisa dapat ditarik beberapa kesimpulan yaitu:

a. Apabila pada kontroler diberi sebuah integrator maka nilai *error steady state* akan hilang, tetapi pemberian nilai K_i juga perlu diperhatikan karena pemberian nilai yang berlebihan dapat menyebabkan sinyal kontrol berosilasi.
b. Semakin besar beban yang diberikan maka semakin besar juga sinyal kontrol yang akan dikeluarkan.
c. Setelah dilihat hasil simulasi terdapat sedikit perbedaan yaitu pada saat implementasi hasil respon yang didapatkan ternyata lebih cepat. Dari data yang didapatkan hasil respon simulasi memiliki nilai *settling time* sekitar 4,5 detik sedangkan respon saat implementasi memiliki *settling time* sekitar 3,6 detik. Dan untuk nilai *rise time* saat simulasi respon membutuhkan waktu sekitar 3,4 detik sedangkan untuk implementasi membutuhkan waktu sekitar 2,6 detik.

5.2 Saran

Apabila ada yang ingin menggunakan *plant* motor BLDC V-1 ini untuk dilakukan penelitian lebih lanjut atau menggunakan kontroler *Fuzzy* PID *Gain Scheduling*, disarankan beberapa hal:

a. *Plant* ini memiliki beberapa parameter yang dapat mengubah-ubah kecepatan motor BLDC V-1. Pengaruh ini bukan hanya dari *input* PWM, tetapi ada beberapa faktor yang penulis temukan saat mengerjakan tugas akhir ini. Diantaranya ketegangan dari *belt* yang mengkopel motor dengan poros piringan, karena apabila kondisi *belt* terlalu tegang maka kecepatan motor BLDC V-1 akan berkurang karena banyaknya gaya yang dibebankan pada motor. Lalu kemiringan dari pijakan *plant*. Lalu kondisi baterai juga dapat mempengaruhi kecepatan motor BLDC V-1.
b. Dalam melakukan implementasi kontroler harus dilihat dulu sinyal kontrol yang dikeluarkan kontroler. Karena respon yang berlebih dapat menyebabkan terjadinya hentakan pada motor.

c. Dalam merancang kontroler *Fuzzy* PID *Gain Scheduling* harus diketahui model dari motor terdahulu karena untuk mendapatkan nilai K_p, K_i, dan K_d tidak bisa didapatkan dengan teori-teori tertentu sehingga harus dilakiukan *try and error* yang terstruktur.

DAFTAR PUSTAKA

[1] Mosavi, Mohamad; Reza .Abdoreza Rahmati dan Alireza Khoshaadat, "Design of Efficient Adaptive Neuro Fuzzy Controller Based on Supervisory Learning Capable for Peed and Torque Control of BLDC Motor," , Iran, 2012.

[2] Abe Dharmawan, *"Pengendalian Motor Vrushless Dengan Metode PWM Sinusoidal Menggunakan ATMEGA 16"*.: Skripsi UI, 2009.

[3] Covey, Greg, "Brushless Basic" <URL: http://www.rcuniverse.com/magazine/article_display.cfm?article_id=1344>, Mei, 2011.

[4] _____, "The Relative Motion Between A Conductor And Magnetic Field Is Used To Generate An Electrical Voltage", <URL:www.boredofstudies.org/wiki/The_relative_motion_between_a_conductor_and_magnetic_field_is_used_to_generate_an_electrical_voltage>, Maret, 2013.

[5] _____, "Electromagnetic Braking", <URL: http://lrrpublic.cli.det.nsw.edu.au/lrrSecure/Sites/Web/physics_explorer/physics/lo/induction_08/induction_08_02.htm>, 2007.

[6] Eliezer, I Putu Giovanni, "Rotary Encoder", <URL: http://www.geyosoft.com/2014/rotary-encoder>, April, 2014.

[7] Elrosa, Ilmiyah, *"*Traction Control pada Parallel Hybrid Electric Vehicle dengan Metode Generalized Predictive Control*"*, *Tugas Akhir*, Jurusan Teknik Elektro, Institut Teknologi Sepuluh Nopember, Surabaya, Bab. 2, 2014.

[8] Ogata, Katsuhiko, *"Modern Control Engineering"*, Prentice Hall, New Jersey, 2002.

[9] Jakoubek, Pavel, "Experimental Identification of Stabile

Nonoscillatory Systems from Step-Responses by Selected Methods," *Konference Studenske Tvuurci-Cinnosti*, 2009.

[10] Wain, Yosep Suban, "Fuzzy Control", <URL: https://asro.wordpress.com/2009/03/27/fuzzy-control>, Maret, 2009.

[11] Passino, Kevin M., Stephen Yurkovich, *"Fuzzy Control"*, Addison Wesley Longman, Inc, Menlo Park, California, 1998.

[12] _____, "Kontroler PID", <URL: http://labkontrol.blogspot.com/2012/09/kontroller-pid_30.html>, September, 2012.

LAMPIRAN

Program Arduino.

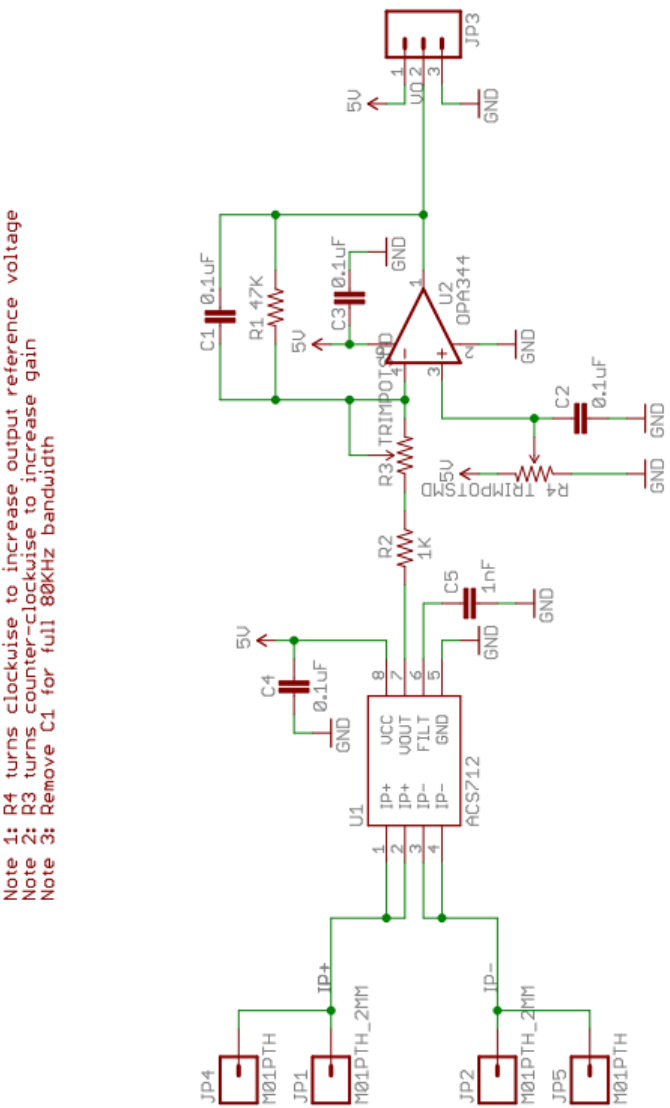
```
#include <Servo.h>
Servo esc;
int setpoint;
int incomingByte;
int percenttrot;
int encoderPin = 7;
unsigned long duration;
unsigned long rpm;
char disp2[1];
char disp3[1];

void setup() {
  Serial.begin(115200);
  pinMode(encoderPin, INPUT);
  esc.attach(9);
  esc.writeMicroseconds(1080);
  delay(2000);
}

void loop() {
  setpoint = map(percenttrot,0,1000,1080,1750);
  esc.writeMicroseconds(setpoint);
  char disp1[10];
  if((setpoint<=1751)&&(setpoint>=1079)){
    sprintf(disp1,"*%4d",percenttrot);
    Serial.print(disp1);
  }
  else{
    setpoint=1080;
    percenttrot=0;
    sprintf(disp1,"*%4d",percenttrot);
    Serial.print(disp1);
  }
```

```
  if (Serial.available()>0){
    percenttrot = Serial.parseInt();
  }
  duration = pulseIn(encoderPin, HIGH, 100000);
  rpm=(79550000/3)/duration;
  if (rpm>4000){
    rpm=0;
  }
  Serial.print('#');
  sprintf(disp2,"%4d",rpm);
  Serial.print(disp2);
  int sensorValue = analogRead(A0);
  int rem = map(sensorValue,0,1023,1,254);
  analogWrite(11,rem);
  int percentrem=map(sensorValue,0,1023,1,999);
  sprintf(disp3,"%3d",percentrem);
  Serial.print('@');
  Serial.print(disp3);
}
```

Skematik Rangkaian Sensor Arus ACS712



RIWAYAT HIDUP

Hudaibiy Hibban lahir di Tangerang tanggal 8 Oktober 1993, merupakan anak ketiga dari Syaiful Anshori dan Fauzia Tanjung. Setelah lulus dari SMA Negeri 86 Jakarta Selatan tahun 2011, melanjutkan studi di Jurusan Teknik Elektro Institut Teknologi Sepuluh Nopember (ITS) Surabaya pada tahun yang sama.